NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

TRANSMISSION CONTROL PROTOCOL AND INTERNET PROTOCOL

Graduation Project
COM-400

Student: Rehan Yasir Saroia

Supervisor: Asst. Prof. Dr Firudin Muradov

Nicosia-2004

ACKNOWLEDGEMENTS

" First of all I would like to thank my ALLAH who helped me during this project and who helps me in every matter of my life. Without the help of ALLAH I was not this caliber to complete my project.

I would like to express my deep appreciation to my advisor Asst. Prof. Dr. Firudin Muradov, who helped me to approach the problems from different perspectives and to define the scope of what I needed to accomplish, his suggestions and observations were extremely helpful throughout this project.

A special note of appreciation and thanks goes to my parents and all the other family members for their unlimited encouragement, prayers, support and love during my life.

I would also like to thank my all friends who encouraged and helped me during my project."

ABSTRACT

Transmission Control Protocol/Internet Protocol (TCP/IP) is an industry-standard suit of protocols designed for Wide Area Networks (WANs). The roots of the TCP/IP can be traced back to the packet switching network experiments conducted by the US Department of Defence Advanced Research Projects Agency (DARPA). IP is a connectionless protocol primarily responsible for addressing and routing packets between hosts, that is, a session is not established before exchanging data. IP is unreliable in that delivery is not guaranteed. An acknowledgement is not required when data is received. where as Transmission Control Protocol (TCP) is responsible for controlling the transmission of data from one host to another host. The TCP/IP utilities include File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), Remote Procedure Call (RPC), Telnet, Simple Network Management Protocol (SNMP), Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP).

TABLE OF CONTENTS

LIST OF ABRIVIATION

ANSI	American National Standard Institute
API	Application Programming Interface
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ASCII	American National Standard Code for Information Interchange
BER	Basic Encoding Rules
DARPA	Defense Advanced Research Projects Agency
DCA	Defense Communications Agency
DCE	Distributed Computing Environment
DCE	Data Circuit-Terminating Equipment (also called Data Communications Equipment)
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISODE	ISO Development Environment
LAN	Local Area Network
MAN	Metropolitan Area Network
NFS	Network File System
NIC	Network Interface Card
OSI	Open Systems Interconnect
PPP	Point-to-Point Protocol
RFC	Request For Comment
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TELNET	Terminal Networking
TFTP	Trivial File Transfer Protocol
TP4	OSI Transport Class 4
TSAP	Transport Service Access Point
UDP	User Datagram Protocol
WAN	Wide Area Network
XNS	Xerox Network Systems

INTRODUCTION

All modern operating systems offer TCP support and most large networks rely on TCP for much of their network traffic. This is the technology for connecting dissimilar systems.

In first chapter, we will see the relationship of OSI and TCP/IP layered architectures, a history of TCP/IP and the Internet, the structure of the Internet, Internet and IP addresses, TCP/IP Components and Different Environments Support by TCP/IP. Using these concepts, we will then move on to look ARP and IP Addresses and The Domain Name System.

The second chapter begins with the TCP/IP configuration files for administrative basics, showing how it configures and the format of its commands in the basic files. The rest of the chapter covers gateway information necessary to configure the TCP/IP on system. We will start in-depth look at the, Setting the Host Name, Loop Back Driver, managing the Address Resolution Protocol. We will cover how to use ifconfig and how it does its task. The function and working of The inetd Daemon, The netstat Command programs in TCP/IP configuration. In the end we will look at the ping Utility and Tracing a TCP/IP based Connection after configuring on system.

The third chapter moves to the setting up a TCP/IP based server and client. First we will look at how to set up a TCP/IP based server in the Windows NT environment. In this we will also discuss The Sample Network, Configuring TCP/IP Software, Configuring Windows NT Server and Testing the Server Configurations. We will also look at the setting up TCP/IP based Windows Client. Here we will look installation, configuration and testing of Windows-Based TCP/IP NetManage's Chameleon.

In fourth chapter, we will see the troubleshooting of TCP/IP. In this we will look how we can troubleshoot the problems usually faced after configuring TCP/IP on any machine. The problems could be in the Network Interface, Network (IP) Layer, TCP and UDP confliction and Application Layer. So we will discuss here Troubleshooting of Network Interface, Network (IP) Layer, TCP and UDP and Application Layer. In last we will have a look in the security in TCP/IP transmission, as Routers can be significant in a network's security plan so by restrict traffic through the router in some manner how can we increase security.

1. INTRODUCTION AND IMPLEMENTATION OF TCP/IP

1.1 Protocols

Computer protocols define the manner in which communications take place. If one computer is sending information to another and they both follow the protocol properly, the message gets through, regardless of what types of machines they are and what operating systems they run. As long as the machines have software that can manage the protocol, communications are possible. Essentially, a computer protocol is a set of rules that coordinates the exchange of information.
Protocols have developed from very simple processes ("I'll send you one character, you send it back, and I'll make sure the two match") to elaborate, complex mechanisms that cover all possible problems and transfer conditions. A task such as sending a message from one coast to another can be very complex when you consider the manner in which it moves. A single protocol to cover all aspects of the transfer would be too large, unwieldy, and overly specialized. Therefore, several protocols have been developed, each handling a specific task.

1.1.1 Protocol Headers

Protocol control information is information about the datagram to which it is attached. This information is usually assembled into a block that is attached to the front of the data it accompanies and is called a header or protocol header. Protocol headers are used for transferring information between layers as well as between machines. When a protocol header is passed to the layer beneath, the datagram including the layer's header is treated as the entire datagram for that receiving layer, which adds its own protocol header to the front. Thus, if a datagram started at the application layer, by the time it reached the physical layer, it would have seven sets of protocol headers on it. These layer protocol headers are used when moving back up the layer structure; they are stripped off as the datagram moves up. An illustration of this is shown in Figure1.1

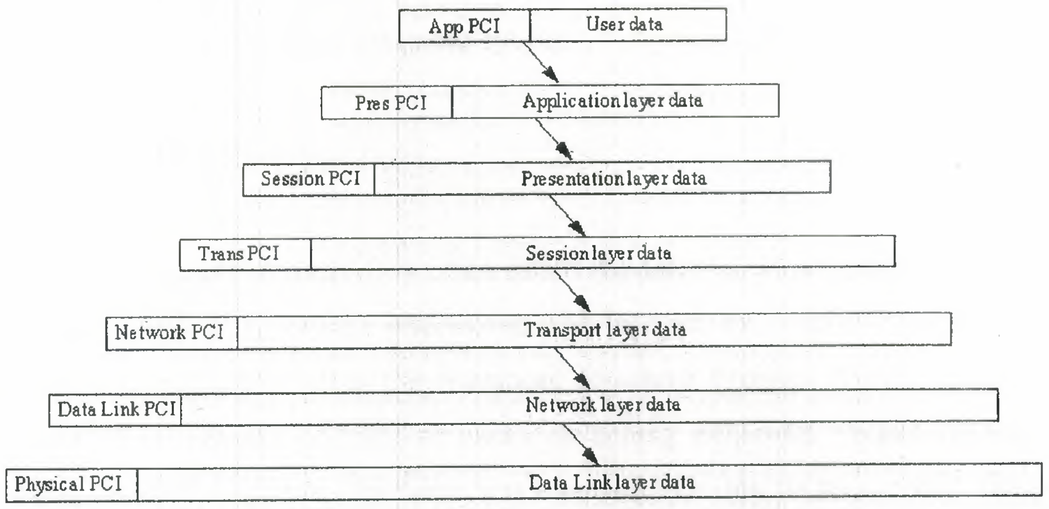


Figure 1.1 layer protocol headers

1.2 TCP/IP

TCP and IP Define as follow:

1.2.1 TCP (Transmission Control Protocol)

TCP is not a piece of software. It is a communications protocol. When you install a TCP stack on your machine, you are installing the TCP layer, and usually a lot more software to provide the rest of the TCP/IP services. TCP is used as a catch-all phrase for TCP/IP in many cases.

1.2.2 IP (Internet Protocol)

The Internet Protocol (IP) is a primary protocol of the OSI model, as well as an integral part of TCP/IP (as the name suggests). Although the word "Internet" appears in the protocol's name, it is not restricted to use with the Internet. It is true that all machines on the Internet can use or understand IP, but IP can also be used on dedicated networks that have no relation to the Internet at all. IP defines a protocol, not a connection. Indeed, IP is a very good choice for any network that needs an efficient protocol for machine-to-machine communications, although it faces some competition from protocols like Novell NetWare's IPX on small to medium local area networks that use NetWare as a PC server operating system

1.3 TCP/IP History

The architecture of TCP/IP is often called the Internet architecture because TCP/IP and the Internet as so closely interwoven The Internet was originally proposed by the precursor of DARPA, called the Advanced Research Projects Agency (ARPA), as a method of testing the viability of packet-switching networks. (When ARPA's focus became military in nature, the name was changed.) During its tenure with the project, ARPA foresaw a network of leased lines connected by switching nodes. The network was called ARPANET, and the switching nodes were called Internet Message Processors, or IMPs.

The ARPANET was initially to be comprised of four IMPs located at the University of California at Los Angeles, the University of California at Santa Barbara, the Stanford Research Institute, and the University of Utah. The original IMPs were to be Honeywell 316 minicomputers.

The contract for the installation of the network was won by Bolt, Beranek, and Newman (BBN), a company that had a strong influence on the development of the network in the following years. The contract was awarded in late 1968, followed by testing and refinement over the next five years.

In 1971, ARPANET entered into regular service. Machines used the ARPANET by connecting to an IMP using the "1822" protocol—so called because that was the number of the technical paper describing the system. During the early years, the purpose and utility of the network was widely (and sometimes heatedly) discussed, leading to refinements and modifications as users requested more functionality from the system.

A commonly recognized need was the capability to transfer files from one machine to another, as well as the capability to support remote logins. Remote logins would enable a user in Santa Barbara to connect to a machine in Los Angeles over the network and function as though he or she were in front of the UCLA machine. The protocol then in use on the network wasn't capable of handling these new functionality requests, so new protocols were continually developed, refined, and tested.

Remote login and remote file transfer were finally implemented in a protocol called the Network Control Program (NCP). Later, electronic mail was added through File Transfer Protocol (FTP). Together with NCP's remote logins and file transfer, this formed the basic services for ARPANET.

By 1973, it was clear that NCP was unable to handle the volume of traffic and proposed new functionality. A project was begun to develop a new protocol. The TCP/IP and gateway architectures were first proposed in 1974. The published article by Cerf and Kahn described a system that provided a standardized application protocol that also used end-to-end acknowledgments.

Neither of these concepts were really novel at the time, but more importantly (and with considerable vision), Cerf and Kahn suggested that the new protocol be independent of the underlying network and computer hardware. Also, they proposed universal connectivity throughout the network. These two ideas were radical in a world of proprietary hardware and software, because they would enable any kind of platform to participate in the network. The protocol was developed and became known as TCP/IP.

A series of RFCs (Requests for Comment, part of the process for adopting new Internet Standards) was issued in 1981, standardizing TCP/IP version 4 for the ARPANET. In 1982, TCP/IP supplanted NCP as the dominant protocol of the growing network, which was now connecting machines across the continent. It is estimated that a new computer was connected to ARPANET every 20 days during its first decade. (That might not seem like much compared to the current estimate of the Internet's size doubling every year, but in the early 1980s it was a phenomenal growth rate.)

During the development of ARPANET, it became obvious that nonmilitary researchers could use the network to their advantage, enabling faster communication of ideas as well as faster physical data transfer. A proposal to the National Science Foundation lead to funding for the Computer Science Network in 1981, joining the military with educational and research institutes to refine the network. This led to the splitting of the network into two different networks in 1984. MILNET was dedicated to unclassified military traffic, whereas ARPANET was left for research and other nonmilitary purposes.

ARPANET's growth and subsequent demise came with the approval for the Office of Advanced Scientific Computing to develop wide access to supercomputers. They created NSFNET to connect six supercomputers spread across the country through T-1 lines (which operated at 1.544 Mbps). The Department of Defense finally declared ARPANET obsolete in 1990, when it was officially dismantled.

1.4 TCP/IP Components

The roles of the many components of the TCP/IP protocol family, it is useful to know what can do over a TCP/IP network. Then, once the applications are understood, the protocols that make it possible are a little easier to comprehend. The following list is not exhaustive but mentions the primary user applications that TCP/IP provides.

1.4.1 Telnet

The Telnet program provides a remote login capability. This lets a user on one machine log onto another machine and act as though he or she were directly in front of the second machine. The connection can be anywhere on the local network or on another network anywhere in the world, as long as the user has permission to log onto the remote system.

Telnet can use when need to perform actions on a machine across the country. This isn't often done except in a LAN or WAN context, but a few systems accessible through the Internet allow Telnet sessions while users play around with a new application or operating-system.

1.4.2 File Transfer Protocol

File Transfer Protocol (FTP) enables a file on one system to be copied to another system. The user doesn't actually log in as a full user to the machine he or she wants to access, as with Telnet, but instead uses the FTP program to enable access. Again, the correct permissions are necessary to provide access to the files.

Once the connection to a remote machine has been established, FTP enables you to copy one or more files to your machine. (The term transfer implies that the file is moved from one system to another but the original is not affected. Files are copied.) FTP is a widely used service on the Internet, as well as on many large LANs and WANs.

1.4.3 Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP) is used for transferring electronic mail. SMTP is completely transparent to the user. Behind the scenes, SMTP connects to remote machines and transfers mail messages much like FTP transfers files. Users are almost never aware of SMTP working, and few system administrators have to bother with it. SMTP is a mostly trouble-free protocol and is in very wide use.

1.4.4 Kerberos

Kerberos is a widely supported security protocol. Kerberos uses a special application called an authentication server to validate passwords and encryption schemes. Kerberos is one of the more secure encryption systems used in communications and is quite common in UNIX.

1.4.5 Domain Name System

Domain Name System (DNS) enables a computer with a common name to be converted to a special network address. For example, a PC called Darkstar cannot be accessed by another machine on the same network (or any other connected network) unless some method of checking the local machine name and replacing the name with the machine's hardware address is available. DNS provides a conversion from the common local name to the unique physical address of the device's network connection.

1.4.6 Simple Network Management Protocol

Simple Network Management Protocol (SNMP) provides status messages and problem reports across a network to an administrator. SNMP uses User Datagram Protocol (UDP) as a transport mechanism. SNMP employs slightly different terms from TCP/IP, working with managers and agents instead of clients and servers (although they mean essentially the same thing). An agent provides information about a device, whereas a manager communicates across a network with agents.

1.4.7 Network File System

Network File System (NFS) is a set of protocols developed by Sun Microsystems to enable multiple machines to access each other's directories transparently. They accomplish this by using a distributed file system scheme. NFS systems are common in large corporate environments, especially those that use UNIX workstations.

1.4.8 Remote Procedure Call

The Remote Procedure Call (RPC) protocol is a set of functions that enable an application to communicate with another machine (the server). It provides for programming functions, return codes, and predefined variables to support distributed computing.

1.4.9 Trivial File Transfer Protocol

Trivial File Transfer Protocol (TFTP) is a very simple, unsophisticated file transfer protocol that lacks security. It uses UDP as a transport. TFTP performs the same task as FTP, but uses a different transport protocol.

1.4.10 Transmission Control Protocol

Transmission Control Protocol (the TCP part of TCP/IP) is a communications protocol that provides reliable transfer of data. It is responsible for assembling data passed from higher-layer applications into standard packets and ensuring that the data is transferred correctly.

1.4.11 User Datagram Protocol

User Datagram Protocol (UDP) is a connectionless-oriented protocol, meaning that it does not provide for the retransmission of datagrams (unlike TCP, which is connection-oriented). UDP is not very reliable, but it does have specialized purposes. If the applications that use UDP have reliability checking built into them, the shortcomings of UDP are overcome.

1.4.12 Internet Protocol

Internet Protocol (IP) is responsible for moving the packets of data assembled by either TCP or UDP across networks. It uses a set of unique addresses for every device on the network to determine routing and destinations.

1.4.13 Internet Control Message Protocol

Internet Control Message Protocol (ICMP) is responsible for checking and generating messages on the status of devices on a network. It can be used to inform other devices of a failure in one particular machine. ICMP and IP usually work together.

1.5 Different Environments Support by TCP/IP

The University of California at Berkeley was given a grant in the early 1980s to modify their UNIX operating system to included support for IP. The BSD4.2 UNIX release already offered support for TCP and IP, as well as the Simple Mail Transfer Protocol (SMTP) and Address Resolution Protocol (ARP), but with DARPA's funds, BSD4.3 was developed to provide more complete support.

The BSD4.2 support for IP was quite good prior to this grant, but it was limited to use in small local area networks only. To increase the capabilities of BSD UNIX's IP support, BSD added retransmission capabilities, Time to Live information, and redirection messages. Other features were added, too, enabling BSD4.3 to work with larger networks, internetworks (connections between different networks), and wide area networks connected by leased lines. This process brought the BSD UNIX system (and its licensees, such as Sun's SunOS) in line with the IP standards used on AT&T UNIX and other UNIX-based platforms.

With the strong support for IP among the UNIX community, it was inevitable that manufacturers of other software operating systems would start to produce software that allowed their machines to interconnect to the UNIX IP system. Most of the drive to produce IP versions for non-UNIX operating systems was not because of the Internet (which hadn't started its phenomenal growth at the time) but the desire to integrate the other operating systems into local area networks that used UNIX servers.

This material examines several hardware and software systems, focusing on the most widely used platforms, and shows the availability of IP (and entire TCP/IP suites) for those machines. Much of this is of interest only if you have the particular platform discussed (DEC VAX users tend not to care about interconnectivity to IBM SNA platforms, for example.

1.5.1 MS-DOS

PCs came onto the scene when TCP/IP was already in common use, so it was not surprising to find interconnection software rapidly introduced. In many ways, the PC was a perfect platform as a stand-alone machine with access through a communications package to other larger systems. The PC was perfect for a client/server environment.

There are many PC-based versions of TCP/IP. The most widely used packages come from FTP Software, The Wollongong Group, and Beame and Whiteside Software Inc. All the packages feature interconnection capabilities to other machines using TCP/IP, and most add other useful features such as FTP and mail routing.

FTP Software's PC/TCP is one of the most widely used. PC/TCP supports the major network interfaces: Packet Driver, IBM's Adapter Support Interface (ASI), Novell's Open Data Link Interface (ODI), and Microsoft/3Com's Network Driver Interface Specification (NDIS). All four LAN interfaces are discussed in more detail in the section titled "Local Area Networks" later today.

The design of PC/TCP covers all seven layers of the OSI model, developed in such a manner that components can be configured as required to support different transport mechanisms and applications. Typically, the Packet Driver, ASI, ODI, or NDIS module has a generic PC/TCP kernel on top of it, with the PC/TCP application on top of that.

PC/TCP enables the software to run both TCP/IP and another protocol, such as DECnet, Novell NetWare, or LAN Manager, simultaneously. This can be useful for enabling a PC to work within a small LAN workgroup, as well as within a larger network, without switching software.

1.5.2 Microsoft Windows

There are several TCP/IP products appearing for Microsoft's Windows 3.x, Windows for Workgroups, and Windows 98. Most of the early packages for Windows 3.x were ports of DOS products. Although these tend to work well, a totally Windows-designed product tends to have a slight edge in terms of integration with the Windows environment. Windows for Workgroups 3.11 has no inherent TCP/IP drives, but several products are available to add TCP/IP suites for this GUI, as well as Windows 3.1 and Windows 3.11.

One Windows 3.x-designed product is NetManage's Chameleon TCP/IP for Windows. Chameleon offers a complete port of TCP/IP and additional software utilities to enable a PC running Windows 3.x to integrate into a TCP/IP network. Chameleon offers terminal emulation, Telnet, FTP, electronic mail, DNS directory services, and NFS capabilities. There are several versions of Chameleon, depending on whether NFS is required.

Windows 98 has TCP/IP drivers included with the distribution software, but they are not loaded by default (NetWare's IPX/SPX is the default protocol for Windows 98). You must install and configure the TCP/IP product as a separate step after installing Windows 98 if you want to use IP on your network

1.5.3 Windows NT

Windows NT is ideally suited for TCP/IP because it is designed to act as a server and gateway. Although Windows NT is not inherently multiuser, it does work well as a TCP/IP access device. Windows NT includes support for the TCP/IP protocols as a network transport, although the implementation does not include all the utilities usually associated with TCP/IP. TCP/IP can be chosen as the default protocol on a Windows NT machine when the operating system is installed.

Among the add-on products available for Windows NT, NetManage's Chameleon32 is a popular package. Similar to the Microsoft Windows version, Chameleon32 offers versions for NFS.

1.5.4 OS/2

IBM's OS/2 platform has a strong presence in corporations because of the IBM reputation and OS/2's solid performance. Not surprisingly, TCP/IP products are popular in these installations, as well. Although OS/2 differs from DOS in many ways, it is possible to run DOS-based ports of TCP/IP software under OS/2. A better solution is to run a native OS/2 application. Several TCP/IP OS/2-native implementations are available, including a TCP/IP product from IBM itself.

1.5.5 Macintosh

Except for versions of UNIX that run on the Macintosh, the Macintosh and UNIX worlds have depended on several different versions of TCP/IP to keep them connected. With many corporations now wanting their investment in Macintosh computers to serve double duty as X terminals onto UNIX workstation, TCP/IP for the Mac has become even more important.

Macintosh TCP products are available in several forms, usually as an add-on application or device driver for the Macintosh operating system. An alternative is Tenon Intersystems' MachTen product line, which enables a UNIX kernel and the Macintosh operating system to coexist on the same machine, providing compatibility between UNIX and the Macintosh file system and Apple events. TCP/IP is part of the MachTen product.

The AppleTalk networking system enables Macs and UNIX machines to interconnect to a limited extent, although this requires installation of AppleTalk software on the UNIX host—something many system administrators are reluctant to do. Also, because AppleTalk is not as fast and versatile as Ethernet and other network transports, this solution is seldom favored.

A better solution is simply to install TCP/IP on the Macintosh using one of several commercial packages available. Apple's own MacTCP software product can perform the basic services but must be coupled with software from other vendors for the higher layer applications. MacTCP also requires a Datagram Delivery Protocol to Internet Protocol (DDP-to-IP) router to handle the sending and receiving of DDP and IP datagrams.

Apple's MacTCP functions by providing the physical through transport layers of the architecture. MacTCP allows for both LocalTalk and Ethernet hardware and supports both IP and TCP, as well as several other protocols. Running on top of MacTCP is the third-party application, which uses MacTCP's function calls to provide the final application for the user. Functions such as Telnet and FTP protocols are supported with add-on software, too.

1.5.6 DEC

Digital Equipment Corporation's minicomputers were for many years a mainstay in scientific and educational research, so an obvious development for DEC and third-party software companies was to introduce IP software. Most DEC machines run either VMS or Ultrix (DEC's licensed version of UNIX). Providing IP capabilities to Ultrix was a matter of duplicating the code developed at Berkeley, but VMS was not designed for IP-type communications, relying instead on DEC's proprietary network software.

DEC's networking software is the Digital Network Architecture (DECnet). The first widely used version was DECnet Phase IV (introduced in 1982), which used industry-standard protocols for the lower layers but was proprietary in the upper layers. The 1987 release of DECnet Phase V provided a combined DECnet IV and OSI system that allowed new OSI protocols to be used within the DECnet environment.

DEC announced the ADVANTAGE-NETWORKS in 1991 as an enhancement of DECnet Phase V, adding support for the Internet Protocols. With the ADVANTAGE-NETWORKS, users could choose between the older, DEC-specific DECnet, OSI, or IP schemes. ADVANTAGE-NETWORKS is DEC's attempt to provide interoperability, providing the DEC-exclusive DECnet system for LAN use, and the TCP/IP and OSI systems for WANs and system interconnection between different hardware types.

Users of VMS systems can connect to the UNIX environment in several ways. The easiest is to use a software gateway between the VMS machine and a UNIX machine. DEC's TCP/IP Services for VMS performs this function, as do several third-party software solutions, such as the Kermit protocol from Columbia University, Wollongong Group's WIN/TCP, and TGV's MultiNet. The advantage of the third-party communications protocol products such as Kermit is that they don't have to be connected to a UNIX machine, because any operating system that supports the communications protocol will work.

ADVANTAGE-NETWORKS users have more options available, many from DEC. Because the protocol is already embedded in the network software, it makes the most sense simply to use it as it comes, if it fits into the existing system architecture. Because of internal conversion software, ADVANTAGE-NETWORKS can connect from a DECnet machine using either the DECnet or the OSI protocols.

1.5.7 IBM's SNA

IBM's Systems Network Architecture (SNA) is in widespread use for both mainframes and minicomputers. Essentially all IBM equipment provides full support for IP and TCP, as well as many other popular protocols. Native IBM software is available for each machine, and several third-party products have appeared (usually at a lower cost than those offered by IBM).

The IBM UNIX version, AIX (which few people know stands for Advanced Interactive Executive), has the TCP/IP software built in, enabling any machine that can run AIX (from workstations to large minicomputers) to interconnect through IP with no additional software. The different versions of AIX have slightly different support, so users should check before blindly trying to connect AIX machines.

For large systems such as mainframes, IBM has the 3172 Interconnect Controller, which sits between the mainframe and a network. The 3172 is a hefty box that handles high-speed traffic between a mainframe channel and the network, off-loading the processing for the communications aspect from the mainframe processor. It can connect to Ethernet or token ring networks and through additional software to DEC's DECnet.

IBM mainframes running either MVS or VM can run software appropriately called TCP/IP for MVS and TCP/IP for VM. These products provide access from other machines running TCP/IP to access the mainframe operating system remotely, usually over a LAN. The software enables the calling machine (the client in a client/server scheme) to act as a 3270-series terminal to MVS or VM. FTP is provided for file transfers with automatic conversion from EBCDIC to ASCII. An interface to PROFS is available. Both TCP/IP software products support SMTP for electronic mail.

1.5.8 Local Area Networks

LANs are an obvious target for TCP/IP, because TCP/IP helps solve many interconnection problems between different hardware and software platforms. To run TCP/IP over a network, the existing network and transport layer software must be replaced with TCP/IP, or the two must be merged together in some manner so that the LAN protocol can carry TCP/IP information within its existing protocol (encapsulation). Whichever solution is taken for the lower layer, a higher layer interface also must be developed, which resides in the equivalent of the data link layer, communicating between the higher layer applications and the hardware. This interface enables the higher layers to be independent of the hardware when using TCP/IP, which many popular LAN operating systems are not currently able to claim.

Three interfaces (which have been mentioned earlier today) are currently in common use. The Packet Driver interface was the first interface developed to meet these needs. 3Com Corporation and Microsoft developed the Network Driver Interface Specification (NDIS) for OS/2 and 3Com's networking software. NDIS provides a driver to communicate with the networking hardware and a protocol driver that acts as the interface to the higher layers. Novell's Open Data Link Interface (ODI) is similar to NDIS.

For single-vendor, PC-based networks, several dedicated TCP/IP packages are available, such as Novell's LAN WorkPlace, designed to enable any NetWare system to connect to a LAN using an interface hardware card and a software driver.

1.6 OSI and TCP/IP

The adoption of TCP/IP didn't conflict with the OSI standards because the two developed concurrently. In some ways, TCP/IP contributed to OSI, and vice-versa. Several important differences do exist, though, which arise from the basic requirements of TCP/IP which are:

- A common set of applications
- Dynamic routing
- Connectionless protocols at the networking level
- Universal connectivity

- Packet-switching

The differences between the OSI architecture and that of TCP/IP relate to the layers above the transport level and those at the network level. OSI has both the session layer and the presentation layer, whereas TCP/IP combines both into an application layer. The requirement for a connectionless protocol also required TCP/IP to combine OSI's physical layer and data link layer into a network level. TCP/IP also includes the session and presentation layers of the OSI model into TCP/IP's application layer. A schematic view of TCP/IP's layered structure compared with OSI's seven-layer model is shown in Figure1.2. TCP/IP calls the different network level elements subnetwork

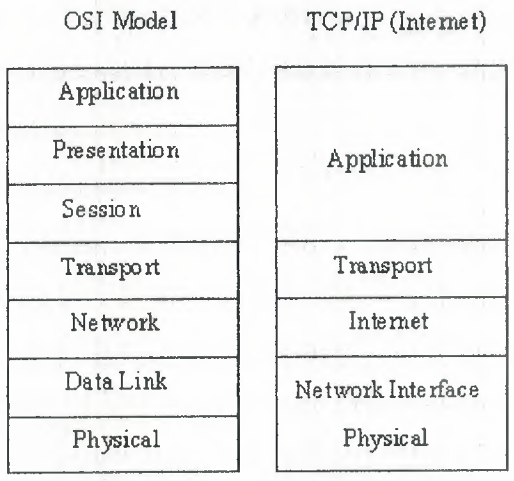


Figure 1.2 OSI's seven-layer

Some fuss was made about the network level combination, although it soon became obvious that the argument was academic, as most implementations of the OSI model combined the physical and link levels on an intelligent controller (such as a network card). The combination of the two layers into a single layer had one major benefit: it enabled a subnetwork to be designed that was independent of any network protocols, because TCP/IP was oblivious to the details. This enabled proprietary, self-contained networks to implement the TCP/IP protocols for connectivity outside their closed systems.

The layered approach gave rise to the name TCP/IP. The transport layer uses the Transmission Control Protocol (TCP) or one of several variants, such as the User Datagram Protocol (UDP). (There are other protocols in use, but TCP and UDP are the most common.) There is, however, only one protocol for the network level—the Internet Protocol (IP). This is what assures the system of universal connectivity, one of the primary design goals.

There is a considerable amount of pressure from the user community to abandon the OSI model (and any future communications protocols developed that conform to it) in favor of TCP/IP. The argument hinges on some obvious reasons:

- TCP/IP is up and running and has a proven record.
- TCP/IP has an established, functioning management body.
- Thousands of applications currently use TCP/IP and its well-documented application programming interfaces.
- TCP/IP is the basis for most UNIX systems, which are gaining the largest share of the operating system market (other than desktop single-user machines such as the PC and Macintosh).
- TCP/IP is vendor-independent.

Arguing rather strenuously against TCP/IP, surprisingly enough, is the US government—the very body that sponsored it in the first place. Their primary argument is that TCP/IP is not an internationally adopted standard, whereas OSI has that recognition. The Department of Defense has even begun to move its systems away from the TCP/IP protocol set. A compromise will probably result, with some aspects of OSI adopted into the still-evolving TCP/IP protocol suite

1.7 TCP/IP and Ethernet

For many people the terms TCP/IP and Ethernet go together almost automatically, primarily for historical reasons, as well as the simple fact that there are more Ethernet-based TCP/IP networks than any other type. Ethernet was originally developed at Xerox's Palo Alto Research Center as a step toward an electronic office communications system, and it has since grown in capability and popularity.

Ethernet is a hardware system providing for the data link and physical layers of the OSI model. As part of the Ethernet standards, issues such as cable type and broadcast speeds are established. There are several different versions of Ethernet, each with a different data transfer rate. The most common is Ethernet version 2, also called 10Base5, Thick Ethernet, and IEEE 802.3 (after the number of the standard that defines the system adopted by the Institute of Electrical and Electronic Engineers). This system has a 10 Mbps rate.

There are several commonly used variants of Ethernet, such as Thin Ethernet (called 10Base2), which can operate over thinner cable (such as the coaxial cable used in cable television systems), and Twisted-Pair Ethernet (10BaseT), which uses simple twisted-pair wires similar to telephone cable. The latter variant is popular for small companies because it is inexpensive, easy to wire, and has no strict requirements for distance between machines.

Ethernet and TCP/IP work well together, with Ethernet providing the physical cabling (layers one and two) and TCP/IP the communications protocol (layers three and four) that is broadcast over the cable. The two have their own processes for packaging information: TCP/IP uses 32-bit addresses, whereas Ethernet uses a 48-bit scheme. The two work together, however, because of one component of TCP/IP called the Address Resolution Protocol (ARP), which converts between the two schemes. (I discuss ARP in more detail later, in the section titled "Address Resolution Protocol.")

Ethernet relies on a protocol called Carrier Sense Multiple Access with Collision Detect (CSMA/CD). To simplify the process, a device checks the network cable to see if anything is currently being sent. If it is clear, the device sends its data. If the cable is busy (carrier detect), the device waits for it to clear. If two devices transmit at the same time (a collision), the devices know because of their constant comparison of the cable traffic to the data in the sending buffer. If a collision occurs, the devices wait a random amount of time before trying again.

1.8 The Internet

As ARPANET grew out of a military-only network to add subnetworks in universities, corporations, and user communities, it became known as the Internet. There is no single network called the Internet, however. The term refers to the collective network of subnetworks. The one thing they all have in common is TCP/IP as a communications protocol.

The organization of the Internet and adoption of new standards is controlled by the Internet Advisory Board (IAB). Among other things, the IAB coordinates several task forces, including the Internet Engineering Task Force (IETF) and Internet Research Task Force (IRTF). In a nutshell, the IRTF is concerned with ongoing research, whereas the IETF handles the implementation and engineering aspects associated with the Internet.

A body that has some bearing on the IAB is the Federal Networking Council (FNC), which serves as an intermediary between the IAB and the government. The FNC has an advisory capacity to the IAB and its task forces, as well as the responsibility for managing the government's use of the Internet and other networks. Because the government was responsible for funding the development of the Internet, it retains a considerable amount of control, as well as sponsoring some research and expansion of the Internet.

1.8.3 The Structure of the Internet

As mentioned earlier, the Internet is not a single network but a collection of networks that communicate with each other through gateways. A gateway (sometimes called a router) is defined as a system that performs relay functions between networks, as shown in Figure1.3. The different networks connected to each other through gateways are often called subnetworks, because they are a smaller part of the larger overall network. This does not imply that a subnetwork is small or dependent on the larger network. Subnetworks are complete networks, but they are connected through a gateway as a part of a larger internetwork, or in this case the Internet.

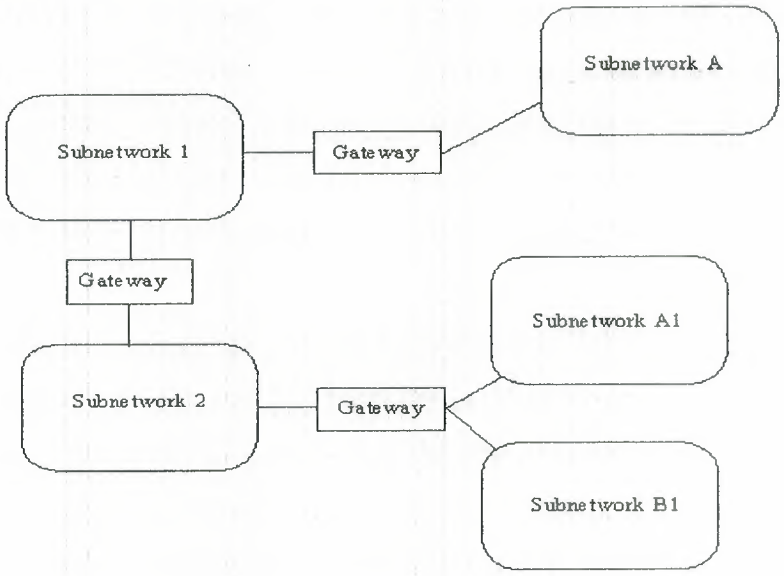


Figure 1.3 Gateway (sometimes called a router)

With TCP/IP, all interconnections between physical networks are through gateways. An important point to remember for use later is that gateways route information packets based on their destination network name, not the destination machine. Gateways are supposed to be completely transparent to the user, which alleviates the gateway from handling user applications (unless the machine that is acting as a gateway is also someone's work machine or a local network server, as is often the case with small networks). Put simply, the gateway's sole task is to receive a Protocol Data Unit (PDU) from either the internetwork or the local network and either route it on to the next gateway or pass it into the local network for routing to the proper user.

Gateways work with any kind of hardware and operating system, as long as they are designed to communicate with the other gateways they are attached to (which in this case means that it uses TCP/IP). Whether the gateway is leading to a Macintosh network, a set of IBM PCs, or mainframes from a dozen different companies doesn't matter to the gateway or the PDUs it handles.

1.8.4 The Internet Layers

Most internetworks, including the Internet, can be thought of as a layered architecture (yes, even more layers!) to simplify understanding. The layer concept helps in the task of developing applications for internetworks. The layering also shows how the different parts of TCP/IP work together. The more logical structure brought about by using a layering process has already been seen in the first chapter for the OSI model, so applying it to the Internet makes sense. Be careful to think of these layers as conceptual only; they are not really physical or software layers as such (unlike the OSI or TCP/IP layers).

It is convenient to think of the Internet as having four layers. This layered Internet architecture is shown in Figure1.4. These layers should not be confused with the architecture of each machine, as described in the OSI seven-layer model. Instead, they are a method of seeing how the internetwork, network, TCP/IP, and the individual machines work together. Independent machines reside in the subnetwork layer at the bottom of the architecture, connected together in a local area network (LAN) and referred to as the subnetwork.

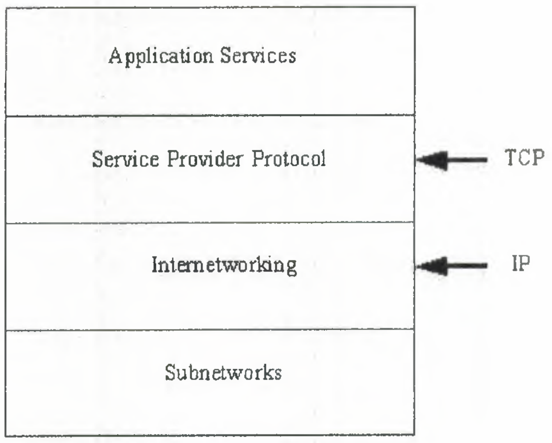


Figure 1.4 Layered Internet Architecture

On top of the subnetwork layer is the internetwork layer, which provides the functionality for communications between networks through gateways. Each subnetwork uses gateways to connect to the other subnetworks in the internetwork. The internetwork layer is where data gets transferred from gateway to gateway until it reaches its destination and then passes into the subnetwork layer. The internetwork layer runs the Internet Protocol (IP).

The service provider protocol layer is responsible for the overall end-to-end communications of the network. This is the layer that runs the Transmission Control Protocol (TCP) and other protocols. It handles the data traffic flow itself and ensures reliability for the message transfer.

The top layer is the application services layer, which supports the interfaces to the user applications. This layer interfaces to electronic mail, remote file transfers, and remote access. Several protocols are used in this layer, many of which you will read about later.

Assume that an application on one machine wants to transfer a datagram to an application on another machine in a different subnetwork. Without all the signals between layers, and simplifying the architecture a little, the process is shown in Figure. The layers in the sending and receiving machines are the OSI layers, with the equivalent Internet architecture layers indicated.

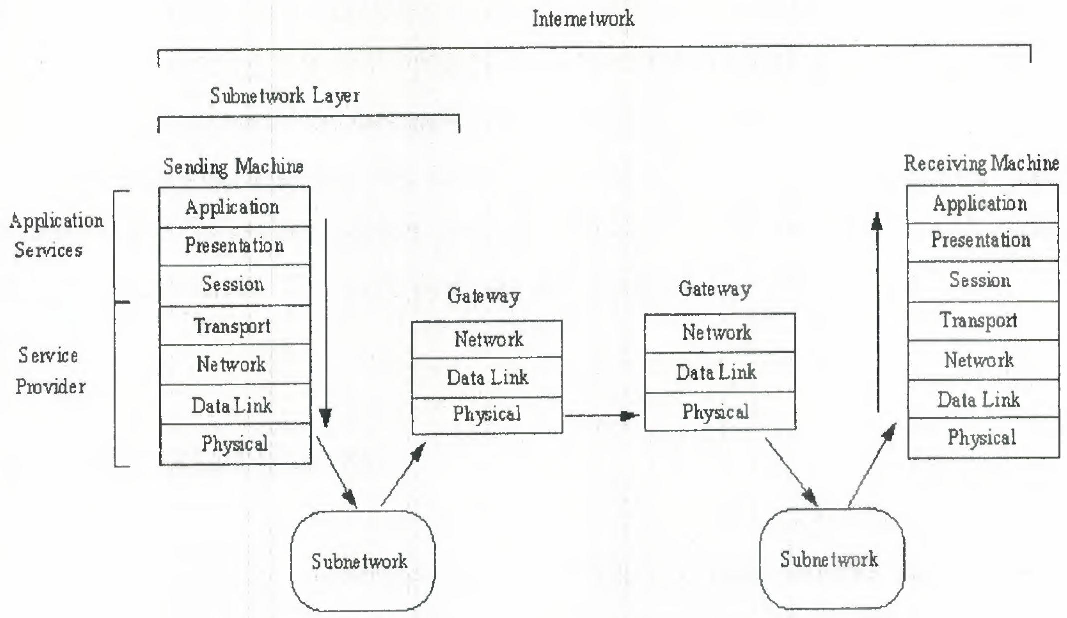


Figure 1.5 Internet Working Architecture

The data is sent down the layers of the sending machine, assembling the datagram with the Protocol Control Information (PCI) as it goes. From the physical layer, the datagram (which is sometimes called a frame after the data link layer has added its header and trailing information) is sent out to the local area network. The LAN routes the information to the gateway out to the internetwork. During this process, the LAN has no concern about the message contained in the datagram. Some networks, however, alter the header information to show, among other things, the machines it has passed through.

From the gateway, the frame passes from gateway to gateway along the internetwork until it arrives at the destination subnetwork. At each step, the gateway analyzes the datagram's header to determine if it is for the subnetwork the gateway leads to. If not, it routes the datagram back out over the internetwork. This analysis is performed in the physical layer, eliminating the need to pass the frame up and down through different layers on each gateway. The header can be altered at each gateway to reflect its routing path.

When the datagram is finally received at the destination subnetwork's gateway, the gateway recognizes that the datagram is at its correct subnetwork and routes it into the LAN and eventually to the target machine. The routing is accomplished by reading the header information. When the datagram reaches the destination machine, it passes up through the layers, with each layer stripping off its PCI header and then passing the result on up. At long last, the application layer on the destination machine processes the final header and passes the message to the correct application.

If the datagram was not data to be processed but a request for a service, such as a remote file transfer, the correct layer on the destination machine would decode the request and route the file back over the internetwork to the original machine. Quite a process!

1.9 Internet Addresses

Network addresses are analogous to mailing addresses in that they tell a system where to deliver a datagram. Three terms commonly used in the Internet relate to addressing: name, address, and route.

A name is a specific identification of a machine, a user, or an application. It is usually unique and provides an absolute target for the datagram. An address typically identifies where the target is located, usually its physical or logical location in a network. A route tells the system how to get a datagram to the address.

You use the recipient's name often, either specifying a user name or a machine name, and an application does the same thing transparently to you. From the name, a network software package called the name server tries to resolve the address and the route, making that aspect unimportant to you. When you send electronic mail, you simply indicate the recipient's name, relying on the name server to figure out how to get the mail message to them.

Using a name server has one other primary advantage besides making the addressing and routing unimportant to the end user: It gives the system or network administrator a lot of freedom to change the network as required, without having to tell each user's machine about any changes. As long as an application can access the name server, any routing changes can be ignored by the application and users.

Naming conventions differ depending on the platform, the network, and the software release, but following is a typical Ethernet-based Internet subnetwork as an example. There are several types of addressing you need to look at, including the LAN system, as well as the wider internetwork addressing conventions.

1.9.3 Subnetwork Addressing

On a single network, several pieces of information are necessary to ensure the correct delivery of data. The primary components are the physical address and the data link address.

1.9.3.1 The Physical Address

Each device on a network that communicates with others has a unique physical address, sometimes called the hardware address. On any given network, there is only one occurrence of each address; otherwise, the name server has no way of identifying the target device unambiguously. For hardware, the addresses are usually encoded into a network interface card, set either by switches or by software. With respect to the OSI model, the address is located in the physical layer.

In the physical layer, the analysis of each incoming datagram (or protocol data unit) is performed. If the recipient's address matches the physical address of the device, the datagram can be passed up the layers. If the addresses don't match, the datagram is ignored. Keeping this analysis in the bottom layer of the OSI model prevents unnecessary delays, because otherwise the datagram would have to be passed up to other layers for analysis.

The length of the physical address varies depending on the networking system, but Ethernet and several others use 48 bits in each address. For communication to occur, two addresses are required: one each for the sending and receiving devices.

The IEEE is now handling the task of assigning universal physical addresses for subnetworks (a task previously performed by Xerox, as they developed Ethernet). For each subnetwork, the IEEE assigns an organization unique identifier (OUI) that is 24 bits long, enabling the organization to assign the other 24 bits however it wants. (Actually, two of the 24 bits assigned as an OUI are control bits, so only 22 bits identify the subnetwork. Because this provides 2^{22} combinations, it is possible to run out of OUIs in the future if the current rate of growth is sustained.)

The format of the OUI is shown in Figure1.6. The least significant bit of the address (the lowest bit number) is the individual or group address bit. If the bit is set to 0, the address refers to an individual address; a setting of 1 means that the rest of the address field identifies a group address that needs further resolution. If the entire OUI is set to 1s, the address has a special meaning which is that all stations on the network are assumed to be the destination

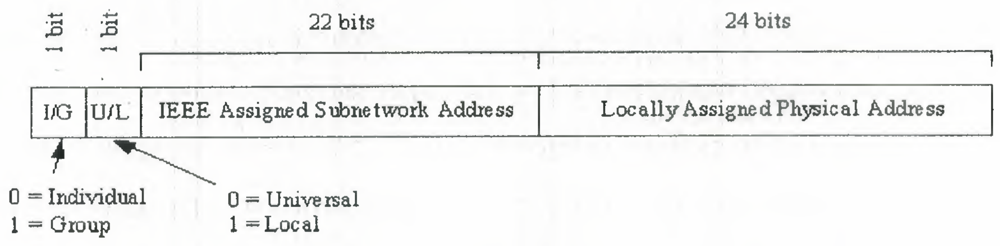


Figure 1.6 Format of OUI

The second bit is the local or universal bit. If set to zero, it has been set by the universal administration body. This is the setting for IEEE-assigned OUIs. If it has a value of 1, the OUI has been locally assigned and would cause addressing problems if decoded as an IEEE-assigned address.

The remaining 22 bits make up the physical address of the subnetwork, as assigned by the IEEE. The second set of 24 bits identifies local network addresses and is administered locally. If an organization runs out of physical addresses (there are about 16 million addresses possible from 24 bits), the IEEE has the capacity to assign a second subnetwork address.

The combination of 24 bits from the OUI and 24 locally assigned bits is called a media access control (MAC) address. When a packet of data is assembled for transfer across an internetwork, there are two sets of MACs: one from the sending machine and one for the receiving machine

1.9.3.2 The Data Link Address

The IEEE Ethernet standards (and several other allied standards) use another address called the link layer address (abbreviated as LSAP for link service access point). The LSAP identifies the type of link protocol used in the data link layer. As with the physical address, a datagram carries both sending and receiving LSAPs. The IEEE also enables a code that identifies the EtherType assignment, which identifies the upper layer protocol (ULP) running on the network (almost always a LAN).

1.9.3.3 Ethernet Frames

The layout of information in each transmitted packet of data differs depending on the protocol, but it is helpful to examine one to see how the addresses and related information are prepended to the data. We use the Ethernet system as an example because of its wide use with TCP/IP. It is quite similar to other systems as well.

A typical Ethernet frame (remember that a frame is the term for a network-ready datagram) is shown in Figure1.7. The preamble is a set of bits that are used primarily to synchronize the communication process and account for any random noise in the first few bits that are sent. At the end of the preamble is a sequence of bits that are the start frame delimiter (SFD), which indicates that the frame follows immediately.

Preamble	Recipient Address	Sender Address	Type	Data	CRC
64 Bits	48 Bits	48 Bits	16 Bits	Variable Length	32 Bits

Figure 1.7 A typical Ethernet frame

Class A addresses are for large networks that have many machines. The 24 bits for the local address (also frequently called the host address) are needed in these cases. The network address is kept to 7 bits, which limits the number of networks that can be identified. Class B addresses are for intermediate networks, with 16-bit local or host addresses and 14-bit network addresses. Class C networks have only 8 bits for the local or host address, limiting the number of devices to 256. There are 21 bits for the network address. Finally, Class D networks are used for multicasting purposes, when a general broadcast to more than one device is required. The lengths of each section of the IP address have been carefully chosen to provide maximum flexibility in assigning both network and local addresses.

IP addresses are four sets of 8 bits, for a total 32 bits. You often represent these bits as separated by a period for convenience, so the IP address format can be thought of as network.local.local.local for Class A or network.network.network.local for Class C. The IP addresses are usually written out in their decimal equivalents, instead of the long binary strings. This is the familiar host address number that network users are used to seeing, such as 147.10.13.28, which would indicate that the network address is 147.10 and the local or host address is 13.28. Of course, the actual address is a set of 1s and 0s. The decimal notation used for IP addresses is properly called dotted quad notation—a bit of trivia for your next dinner party.

The IP addresses can be translated to common names and letters. This can pose a problem, though, because there must be some method of unambiguously relating the physical address, the network address, and a language-based name (such a tpci_ws_4 or bobs_machine). The section later in this chapter titled "The Domain Name System" looks at this aspect of address naming.

From the IP address, a network can determine if the data is to be sent out through a gateway. If the network address is the same as the current address (routing to a local network device, called a direct host), the gateway is avoided, but all other network addresses are routed to a gateway to leave the local network (indirect host). The gateway receiving data to be transmitted to another network must then determine the routing from the data's IP address and an internal table that provides routing information.

The address applies to all addresses on the network. The same rule applies to IP addresses, so that an IP address of 32 1s is considered a broadcast message to all networks and all devices. It is possible to broadcast to all machines in a network by altering the local or host address to all 1s, so that the address 147.10.255.255 for a Class B network (identified as network 147.10) would be received by all devices on that network (255.255 being the local addresses composed of all 1s), but the data would not leave the network.

There are two contradictory ways to indicate broadcasts. The later versions of TCP/IP use 1s, but earlier BSD systems use 0s. This causes a lot of confusion. All the devices on a network must know which broadcast convention is used; otherwise, datagrams can be stuck on the network forever!

A slight twist is coding the network address as all 0s, which means the originating network or the local address being set to 0s, which refers to the originating device only (usually used only when a device is trying to determine its IP address). The all-zero network address format is used when the network IP address is not known but other devices on the network can still interpret the local address. If this were transmitted to another network, it could cause confusion! By convention, no local device is given a physical address of 0.

It is possible for a device to have more than one IP address if it is connected to more than one network, as is the case with gateways. These devices are called multihomed, because they have a unique address for each network they are connected to. In practice, it is best to have a dedicate machine for a multihomed gateway; otherwise, the applications on that machine can get confused as to which address they should use when building datagrams.

Two networks can have the same network address if they are connected by a gateway. This can cause problems for addressing, because the gateway must be able to differentiate which network the physical address is on. This problem is looked at again in the next section, showing how it can be solved

1.10 IP Addresses

TCP/IP uses a 32-bit address to identify a machine on a network and the network to which it is attached. IP addresses identify a machine's connection to the network, not the machine itself—an important distinction. Whenever a machine's location on the network changes, the IP address must be changed, too. The IP address is the set of numbers many people see on their workstations or terminals, such as 127.40.8.72, which uniquely identifies the device.

IP (or Internet) addresses are assigned only by the Network Information Center (NIC), although if a network is not connected to the Internet, that network can determine its own numbering. For all Internet accesses, the IP address must be registered with the NIC.

There are four formats for the IP address, with each used depending on the size of the network. The four formats, called Class A through Class D, are shown in Figure1.8. The class is identified by the first few bit sequences, shown in the figure as one bit for Class A and up to four bits for Class D. The class can be determined from the first three (high-order) bits. In fact, in most cases, the first two bits are enough, because there are few Class D networks

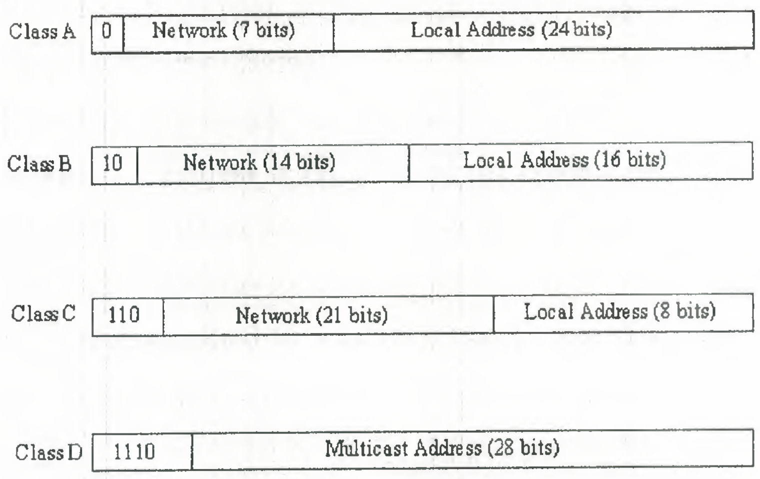


Figure 1.8 Four Formats

Class A addresses are for large networks that have many machines. The 24 bits for the local address (also frequently called the host address) are needed in these cases. The network address is kept to 7 bits, which limits the number of networks that can be identified. Class B addresses are for intermediate networks, with 16-bit local or host addresses and 14-bit network addresses. Class C networks have only 8 bits for the local or host address, limiting the number of devices to 256. There are 21 bits for the network address. Finally, Class D networks are used for multicasting purposes, when a general broadcast to more than one device is required. The lengths of each section of the IP address have been carefully chosen to provide maximum flexibility in assigning both network and local addresses.

IP addresses are four sets of 8 bits, for a total 32 bits. You often represent these bits as separated by a period for convenience, so the IP address format can be thought of as network.local.local.local for Class A or network.network.network.local for Class C. The IP addresses are usually written out in their decimal equivalents, instead of the long binary strings. This is the familiar host address number that network users are used to seeing, such as 147.10.13.28; which would indicate that the network address is 147.10 and the local or host address is 13.28. Of course, the actual address is a set of 1s and 0s. The decimal notation used for IP addresses is properly called dotted quad notation—a bit of trivia for your next dinner party.

The IP addresses can be translated to common names and letters. This can pose a problem, though, because there must be some method of unambiguously relating the physical address, the network address, and a language-based name (such a tpci_ws_4 or bobs_machine). The section later in this chapter titled "The Domain Name System" looks at this aspect of address naming.

From the IP address, a network can determine if the data is to be sent out through a gateway. If the network address is the same as the current address (routing to a local network device, called a direct host), the gateway is avoided, but all other network addresses are routed to a gateway to leave the local network (indirect host). The gateway receiving data to be transmitted to another network must then determine the routing from the data's IP address and an internal table that provides routing information.

As mentioned, if an address is set to all 1s, the address applies to all addresses on the network. (See the previous section titled "Physical Addresses.") The same rule applies to IP addresses, so that an IP address of 32 1s is considered a broadcast message to all networks and all devices. It is possible to broadcast to all machines in a network by altering the local or host address to all 1s, so that the address 147.10.255.255 for a Class B network (identified as network 147.10) would be received by all devices on that network (255.255 being the local addresses composed of all 1s), but the data would not leave the network.

There are two contradictory ways to indicate broadcasts. The later versions of TCP/IP use 1s, but earlier BSD systems use 0s. This causes a lot of confusion. All the devices on a network must know which broadcast convention is used; otherwise, datagrams can be stuck on the network forever!

A slight twist is coding the network address as all 0s, which means the originating network or the local address being set to 0s, which refers to the originating device only (usually used only when a device is trying to determine its IP address). The all-zero network address format is used when the network IP address is not known but other devices on the network can still interpret the local address. If this were transmitted to another network, it could cause confusion! By convention, no local device is given a physical address of 0.

It is possible for a device to have more than one IP address if it is connected to more than one network, as is the case with gateways. These devices are called multihomed, because they have a unique address for each network they are connected to. In practice, it is best to have a dedicate machine for a multihomed gateway; otherwise, the applications on that machine can get confused as to which address they should use when building datagrams.

Two networks can have the same network address if they are connected by a gateway. This can cause problems for addressing, because the gateway must be able to differentiate which network the physical address is on. This problem is looked at again in the next section, showing how it can be solved.

1.11 Address Resolution Protocol

Determining addresses can be difficult because every machine on the network might not have a list of all the addresses of the other machines or devices. Sending data from one machine to another if the recipient machine's physical address is not known can cause a problem if there is no resolution system for determining the addresses. Having to constantly update a table of addresses on each machine would be a network administration nightmare. The problem is not restricted to machine addresses within a small network, because if the remote destination network addresses are unknown, routing and delivery problems will also occur.

The Address Resolution Protocol (ARP) helps solve these problems. ARP's job is to convert IP addresses to physical addresses (network and local) and in doing so, eliminate the need for applications to know about the physical addresses. Essentially, ARP is a table with a list of the IP addresses and their corresponding physical addresses. The table is called an ARP cache. The layout of an ARP cache is shown in Figure1.9. Each row corresponds to one device, with four pieces of information for each device:

	IF INDEX	PHYSICAL ADDRESS	IP ADDRESS	TYPE
Entry 1				
Entry 2				
Entry 3				
Entry n				

Figure 1.9 Layout of an ARP cache

- IF Index: The physical port (interface)
- Physical Address: The physical address of the device
- IP Address: The IP address corresponding to the physical address
- Type: The type of entry in the ARP cache

1.11.3Mapping Types

The mapping type is one of four possible values indicating the status of the entry in the ARP cache. A value of 2 means the entry is invalid; a value of 3 means the mapping is dynamic (the entry can change); a value of 4 means static (the entry doesn't change); and a value of 1 means none of the above.

When the ARP receives a recipient device's IP address, it searches the ARP cache for a match. If it finds one, it returns the physical address. If the ARP cache doesn't find a match for an IP address, it sends a message out on the network. The message, called an ARP request, is a broadcast that is received by all devices on the local network. (You might remember that a broadcast has all 1s in the address.) The ARP request contains the IP address of the intended recipient device. If a device recognizes the IP address as belonging to it, the device sends a reply message containing its physical address back to the machine that generated the ARP request, which places the information into its ARP cache for future use. In this manner, the ARP cache can determine the physical address for any machine based on its IP address.

Whenever an ARP request is received by an ARP cache, it uses the information in the request to update its own table. Thus, the system can accommodate changing physical addresses and new additions to the network dynamically without having to generate an ARP request of its own. Without the use of an ARP cache, all the ARP requests and replies would generate a lot of network traffic, which can have a serious impact on network performance. Some simpler network schemes abandon the cache and simply use broadcast messages each time. This is feasible only when the number of devices is low enough to avoid network traffic problems.

The layout of the ARP request is shown in Figure1.10. When an ARP request is sent, all fields in the layout are used except the Recipient Hardware Address (which the request is trying to identify). In an ARP reply, all the fields are used.

Hardware Type (16 bits)	
Protocol Type (16 bits)	
Hardware Address Length	Protocol Address Length
Operation Code (16 bits)	
Sender Hardware Address	
Sender IP Address	
Recipient Hardware Address	
Recipient IP Address	

Figure 1.10 layout of the ARP request

This layout, which is combined with the network system's protocols into a protocol data unit (PDU), has several fields. The fields and their purposes are as follows:

- Hardware Type: The type of hardware interface
- Protocol Type: The type of protocol the sending device is using
- Hardware Address Length: The length of each hardware address in the datagram, given in bytes
- Protocol Address Length: The length of the protocol address in the datagram, given in bytes
- Operation Code (Opcode): The Opcode indicates whether the datagram is an ARP request or an ARP reply. If the datagram is a request, the value is set to 1. If it is a reply, the value is set to 2.
- Sender Hardware Address: The hardware address of the sending device
- Sender IP Address: The IP address of the sending device
- Recipient IP Address: The IP Address of the recipient
- Recipient Hardware Address: The hardware address of the recipient device

Some of these fields need a little more explanation to show their legal values and field usage. The following sections describe these fields in more detail.

1.11.4The Hardware Type Field

The hardware type identifies the type of hardware interface. Legal values are as follows:

Type	Description
1	Ethernet
2	Experimental Ethernet
3	X.25
4	Proteon ProNET (Token Ring)
5	Chaos
6	IEEE 802.X
7	ARCnet

1.11.5 The Protocol Type Field

The protocol type identifies the type of protocol the sending device is using. With TCP/IP, these protocols are usually an EtherType, for which the legal values are as follows:

Decimal	Description
512	XEROX PUP
513	PUP Address Translation
1536	XEROX NS IDP
2048	Internet Protocol (IP)
2049	X.75
2050	NBS
2051	ECMA
2052	Chaosnet

2053	X.25 Level 3
2054	Address Resolution Protocol (ARP)
2055	XNS
4096	Berkeley Trailer
21000	BBN Simnet
24577	DEC MOP Dump/Load
24578	DEC MOP Remote Console
24579	DEC DECnet Phase IV
24580	DEC LAT
24582	DEC
24583	DEC
32773	HP Probe
32784	Excelan
32821	Reverse ARP
32824	DEC LANBridge
32823	AppleTalk

If the protocol is not EtherType, other values are allowed.

1.12 ARP and IP Addresses

Two (or more) networks connected by a gateway can have the same network address. The gateway has to determine which network the physical address or IP address corresponds with. The gateway can do this with a modified ARP, called the Proxy ARP (sometimes called Promiscuous ARP). A proxy ARP creates an ARP cache consisting of entries from both networks, with the gateway able to transfer datagrams from one network to the other. The gateway has to manage the ARP requests and replies that cross the two networks.

An obvious flaw with the ARP system is that if a device doesn't know its own IP address, there is no way to generate requests and replies. This can happen when a new device (typically a diskless workstation) is added to the network. The only address the device is aware of is the physical address set either by switches on the network interface or by software. A simple solution is the Reverse Address Resolution Protocol (RARP), which works the reverse of ARP, sending out the physical address and expecting back an IP address. The reply containing the IP address is sent by an RARP server, a machine that can supply the information. Although the originating device sends the message as a broadcast, RARP rules stipulate that only the RARP server can generate a reply. (Many networks assign more than one RARP server, both to spread the processing load and to act as a backup in case of problems.)

1.13 The Domain Name System

Instead of using the full 32-bit IP address, many systems adopt more meaningful names for their devices and networks. Network names usually reflect the organization's name (such as tpci.com and bobs_cement). Individual device names within a network can range from descriptive names on small networks (such as tims_machine and laser_1) to more complex naming conventions on larger networks (such as hpws_23 and tpci704). Translating between these names and the IP addresses would be practically impossible on an Internet-wide scale.

To solve the problem of network names, the Network Information Center (NIC) maintains a list of network names and the corresponding network gateway addresses. This system grew from a simple flat-file list (which was searched for matches) to a more complicated system called the Domain Name System (DNS) when the networks became too numerous for the flat-file system to function efficiently.

DNS uses a hierarchical architecture, much like the UNIX filesystem. The first level of naming divides networks into the category of subnetworks, such as com for commercial, mil for military, edu for education, and so on. Below each of these is another division that identifies the individual subnetwork, usually one for each organization. This is called the domain name and is unique. The organization's system manager can further divide the company's subnetworks as desired, with each network called a subdomain. For example, the system merlin.abc_corp.com has the domain name abc_corp.com, whereas the network merlin.abc_corp is a subdomain of merlin.abc_corp.com. A network can be identified with an absolute name (such as merlin.abc_corp.com) or a relative name (such as merlin) that uses part of the complete domain name.

Seven first-level domain names have been established by the NIC so far. These are as follows:

.arpa	An ARPANET-Internet identification
.com	Commercial company
.edu	Educational institution
.gov	Any governmental body
.mil	Military
.net	Networks used by Internet Service Providers
.org	Anything that doesn't fall into one of the other categories

The NIC also allows for a country designator to be appended. There are designators for all countries in the world, such as .ca for Canada and .uk for the United Kingdom. DNS uses two systems to establish and track domain names. A name resolver on each network examines information in a domain name. If it can't find the full IP address, it queries a name server, which has the full NIC information available. The name resolver tries to complete the addressing information using its own database, which it updates in much the same manner as the ARP system (discussed earlier) when it must query a name server. If a queried name server cannot resolve the address, it can query another name server, and so on, across the entire internetwork.

There is a considerable amount of information stored in the name resolver and name server, as well as a whole set of protocols for querying between the two. The details, luckily, are not important to an understanding of TCP/IP, although the overall concept of the address resolution is important when understanding how the Internet translates between domain names and IP addresses

2. TCP/IP CONFIGURATION AND ADMINISTRATION BASICS

2.1 Configuration Files

Several files are involved in the complete specification of network addresses and configuration for TCP/IP. I use a UNIX system standard here in this chapter, although a few other operating systems are mentioned as appropriate. Other operating systems use different filenames, but the purpose of the files is usually the same.

UNIX allows comments on every line of these configuration files, as long as they are prefaced by a pound sign (#). If we see this character in our own system's configuration files, we should note that it is not part of an entry. With many operating systems, the default configuration files have many entries, most of which are commented out until the system administrator removes the comments.

We might not be able to examine the files or run the utilities mentioned in this chapter because of security restrictions. If we edit the configuration files, make sure you do not make any unintentional changes! Make backups of all the files before you make any changes to your systems.

2.1.1 Symbolic Machine Names: /etc/hosts

Whenever a symbolic name is used as a target address by an application, there must be some method to resolve that name into a network address. An ASCII file is commonly used with the symbolic names matched to network addresses. This does not apply when the Yellow Pages (YP), Network Information Services (NIS), or the Domain Name Server (DNS) is used; they use their own configuration files.

On UNIX systems, the file /etc/hosts is used to hold the network addresses, as well as one special connection called the loopback (which is examined later in this chapter in the section titled "The Loopback Driver"). The loopback connection address is usually listed as the machine name loopback or localhost.

The file /etc/hosts consists of the network address in one column separated from the symbolic name in another. The network addresses can be specified in decimal, octal, or hexadecimal format (although decimal is the most common). More than one symbolic name can be specified on a line by separating the names with either space characters or tabs. The /etc/hosts file can be as long as necessary to contain all the symbolic names used on the local machine; they do not need to be presented in any order. A sample UNIX /etc/hosts file is as follows:

```
# network host addresses
127.0.0.1        localhost local tpci_server
157.40.40.1       tpci_sco1
157.40.40.2       tpci_sco2
157.40.40.3       tpci_hpws1
157.40.40.0       tpci_server tpci_main tpci
47.80.157.36       bnr.ca BNR bnr
191.13.123.4       kitty_cat
205.150.89.1       roy_maclean big_roy
210.24.47.128       bobs_machine
```

The file is made up of two columns. The first column gives the IP address of a machine, and the second (separated by one or more whitespace characters) gives the machine's name. If several names can be used to identify the remote machine, they are listed on the same line, separated by whitespace. For example, the remote machine with IP address 205.150.89.1 can be addressed as either roy_maclean or big_roy. Whenever either of those names is used in a command (such as an FTP or Telnet application), this file is used to match to the proper IP address.

A system or network administrator can update the /etc/hosts file at any time, and changes are effective immediately (so the machine doesn't have to be rebooted to effect the changes). Whenever a symbolic name is specified by a user or an application, the /etc/hosts file is always searched first for a matching name, and the proper address is read from the same line.

Most TCP/IP implementations on other platforms have a similar type of file to resolve IP addresses from symbolic names. NetManage ChameleonNFS running on a Windows 3.x machine, for example, uses a Host Table to match names and IP addresses. The Host Table, shown in Figure 2.1, is a graphical front-end to a file equivalent to /etc/hosts on a UNIX machine.

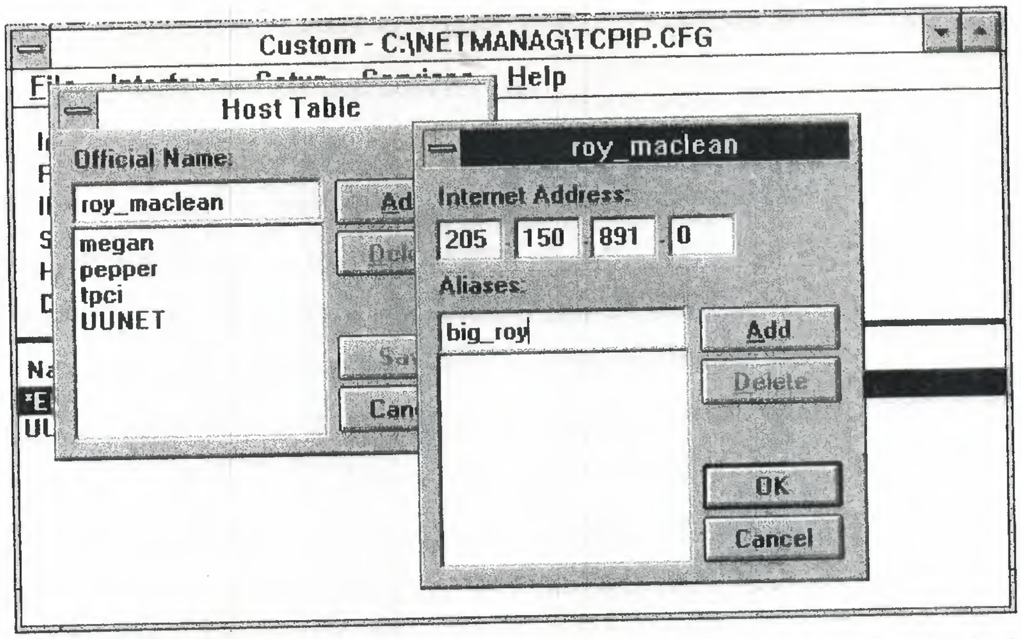


Figure 2.1 Graphical Front-end to a file equivalent to /etc/hosts on a UNIX Machine

2.1.2 Network Names: /etc/networks

Networks can be addressed by a symbolic name, just as machines are. To resolve the network names, another file is used that contains the corresponding network address. Typically, this file isn't accessed often, because few users want to address an entire network within their application. The network name resolution file's most common use is to specify the local network's name.

UNIX systems usually use the file /etc/networks to specify symbolic network names. The format of the file provides a network symbolic name, its network address, and any alias that might be used, in much the same format as the /etc/hosts table is used for specific machines. A sample /etc/networks file is shown here:

```
# local network names
tpci      146.1       tpci_network  tpci_local
bnr       47.80       BNR bnr.ca
tmn       123.2.21
unique    89.123.23   UNIQUE
sco       132.147     SCO
loopback  127         localhost
```

The /etc/networks file layout is a little different from /etc/hosts in that the usual network name is given in the first column, followed by the IP network address, then any aliases.

The last entry in this example file gives the loopback name. The first entry specifies the local machine name, its network address, and any name variants. Using this file, an application that wanted to reach the network called UNIQUE could use that name and let the operating system resolve it to the IP network address 89.123.23.

Many implementations of TCP/IP on other platforms don't bother with a network name resolution file like this. Part of the reason is that the /etc/networks file has little use on a UNIX platform, and many single-user operating systems don't require the type of versatility a multi user operating system like UNIX must supply to an entire network.

2.1.3 Network Protocols: /etc/protocols

Protocol numbers are used to identify the transport protocol to the receiving machine to enable proper decoding of the information within the datagram. With TCP/IP, the protocol number is embedded in the Internet Protocol header. A configuration file is usually used to identify all the transport protocols available on the system and their respective protocol numbers.

UNIX systems use the /etc/protocols file for this purpose. Usually, this file is not modified by the administrator but is maintained by the system and updated automatically as part of the installation procedure when new TCP/IP software or services are added. The /etc/protocols file contains the protocol name, its number, and any alias that might be used for that protocol. A sample /etc/protocols file is shown here:

```
# Internet (IP) protocols
ip      0     IP      # internet protocol, pseudo protocol number
icmp    1     ICMP    # internet control message protocol
igmp    2     IGMP    # internet group management protocol
ggp     3     GGP     # gateway-gateway protocol
tcp     6     TCP     # transmission control protocol
egp     8     EGP     # Exterior-Gateway Protocol
pup     12    PUP     # PARC universal packet protocol
udp     17    UDP     # user datagram protocol
```

```
hello  63     HELLO   # HELLO Routing Protocol
ospf   89     OSPF    # Open Shortest Path First Routing Protocol
```

In this /etc/protocols file, the IP protocol is assigned protocol 0, and TCP is protocol 6. The values in this table should not be changed from their default values except when special network conditions mandate a change. If new TCP/IP services are added to the UNIX system this file resides on, new entries are made to this file by the application installation routine.

There are usually no equivalents of the /etc/protocols file on other operating systems because they assume that the standard transport number is used for each protocol.

2.1.4 Network Services: /etc/services

The final common configuration file used on most UNIX systems identifies the existing network services. As with the /etc/protocols file, this file is not usually modified by an administrator but is maintained by software as it is installed or configured.

The UNIX network services file is /etc/services. The file is in ASCII format consisting of the service name, a port number, and the protocol type. The port number and protocol type are separated by a slash. The port numbers for TCP/IP usually follow the conventions mentioned in the previous chapters. Any optional service alias names follow after the port numbers. A short extract from a sample /etc/services file (the file is usually quite lengthy) is shown here:

```
# network services
echo     7/tcp
echo     7/udp
discard  9/tcp   sink  null
discard  9/udp   sink  null
ftp      21/tcp
telnet   23/tcp
smtp     25/tcp   mail mailx
tftp     69/udp
# specific services
```

```
login   513/tcp
who     513/udp   whod
```

2.2 Setting the Host Name

TCP/IP requires that each machine on the network have an IP address. Usually, each machine also has a unique symbolic name; otherwise, the IP address must be used for all connections to that machine. Most operating systems have a simple program that identifies the name of the local machine. UNIX systems have the utility hostname for this purpose, as well as the uname program, which can give the node name with the command uname -n. The uname utility is usually supported in System V and compatible operating systems only.

The host name is sometimes saved in a separate file that is read when the operating system starts up, or it can be read from one of the configuration files mentioned previously. The hostname is used by most protocols on the system and by many TCP/IP applications, so it is important for proper system operation. The host name can sometimes be changed by editing the system file that contains the name and then rebooting the machine, although many operating systems provide a utility program to ensure that this process is performed correctly.

On many UNIX systems, the hostname and uname commands echo back the local machine name, as the following sample session shows:

```
$ hostname
tpci_sco4.tpci.com
$ uname -n
tpci_sco4
```

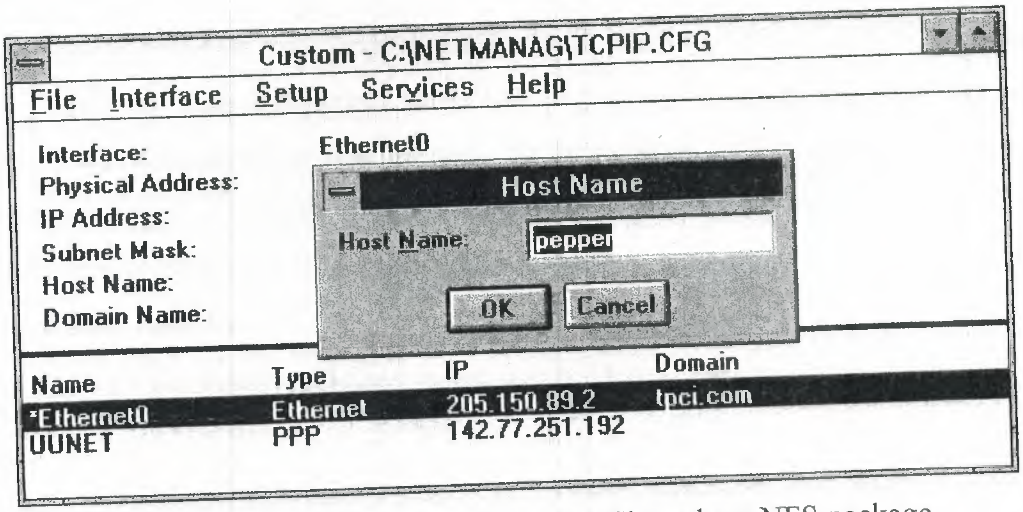
On the SCO UNIX system used in this example, the hostname command returns the fully qualified domain name, whereas the uname command provides the local machine name only. On a Hewlett-Packard workstation running HP-UX, both commands return only the local machine name. The exact behavior of the hostname and uname commands is therefore quite dependent on the implementation.

On a Linux system, for example, the hostname command can be used to not only show the current host name setting but also to change it when used with the -S (for set) option. For example, the command

```
hostname -S willow.tree.com
```

changes the local fully qualified domain name to willow.tree.com. Not all versions of Linux support the -S option of the hostname command.

Most TCP/IP suites for other operating systems use a simpler method of setting the host name. For example, on a Windows 3.x machine the NetManage ChameleonNFS package uses the dialog shown in Figure 2.2 to quickly set the host name.


Custom - C:\NETMANAG\TCPIP.CFG

Interface: Ethernet0
Physical Address:
IP Address:
Subnet Mask:
Host Name:
Domain Name:

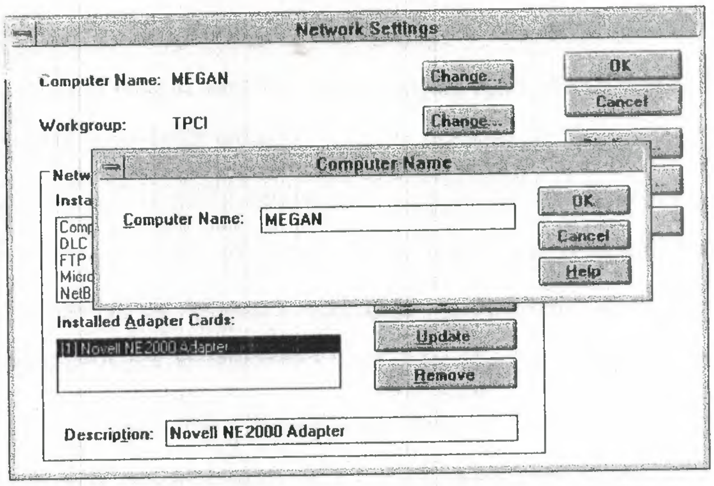
Host Name

Host Name: pepper

Name	Type	IP	Domain
*Ethernet0	Ethernet	205.150.89.2	tpci.com
UUNET	PPP	142.77.251.192	

Figure 2.2 Dialog use by Net Manage Chameleon NFS package

Windows NT has TCP/IP services built into the basic distribution. On a Windows NT system, the host name is specified through the Network dialog opened from the Control Panel, as shown in Figure 2.3. Both the Windows NT and Windows 3.x systems enable a change in the host name to be made effective immediately, although a system reboot is recommended to clear all configuration information held in memory.



Computer Name: MEGAN

Workgroup: TPCI

Computer Name: MEGAN

Installed Adapter Cards: [1] Novell NE2000 Adapter

Description: Novell NE2000 Adapter

Figure 2.3 Open Network dialog

A potential problem can occur when the local machine is multihomed, or based in several networks with a different name and IP address for each network. The single name in the configuration file in such an installation might not provide enough information to permit proper routing over all the connected networks. This problem is seldom encountered, but it does require the system administrator to set the hostname for each network carefully.

Aside from the simple machine name query shown, the hostname system is a full protocol that enables access to the Network Information Center (NIC) tables to verify addresses and obtain information about the network, gateways, and hosts. It uses TCP port number 101 to connect to the NIC. This type of access is usually restricted to the network administrator.

2.3 The Loopback Driver

The loopback driver is probably the most fundamental and often-used diagnostic available to an administrator. A loopback driver acts as a virtual circuit, enabling outgoing information to be immediately rerouted back to an input. This enables testing of the machine's circuits by eliminating any external influences, such as the network itself, gateways, or remote machines. By convention, each machine uses the IP address 127.0.0.1 for the loopback driver (also called the localhost IP address).

Every system should have a loopback driver in place whether the machine is on a network or not. This is because some applications insist on having an IP address they can access to function properly. Many license servers on a UNIX machine have this requirement, for example. Although the need for a loopback driver isn't important for non-networked Windows and similar operating system machines, a loopback driver is always installed with a TCP/IP suite.

Loopback drivers are usually embedded as part of the operating system kernel, or sometimes as an add-on utility program. Most multiuser systems employ an embedded loopback driver. UNIX is a good example: within the kernel is a device driver specifically designed to act as a loopback driver. The loopback driver is almost always added automatically when the operating system is installed, but a few UNIX-based operating systems, including several versions of Linux, don't perform this function, and the loopback driver must be added manually by the system administrator. As previously mentioned, several configuration files on the system contain the address of the loopback's connection, such as /etc/hosts.

Using the loopback driver to reroute the output stream, the network interface card (usually an Ethernet card) is bypassed. *The loopback driver is useful for testing TCP/IP* software installations, because it immediately shows any problems with the local configuration. This can be done before the machine is physically connected to the network or even before the networking hardware and software are installed. For example, you can use the loopback driver to test your TCP/IP configuration before it is connected to a network by using the ping command with the localhost name or IP address, as the following example shows:

```
# ping -c5 localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from localhost (127.0.0.1): icmp_seq=0 ttl=64 time=10 ms
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0 ms
--- localhost ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0/2/10 ms
# ping -c5 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from localhost (127.0.0.1): icmp_seq=0 ttl=64 time=0 ms
```

```
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0 ms
--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms
```

In the preceding example I used the ping command with the -c option to specify five pings, first with the localhost name (which /etc/hosts resolves to the IP address 127.0.0.1) and then with the IP address itself. If either command had failed, it would indicate a problem with either the /etc/hosts file (if the name localhost could not be resolved) or with the TCP/IP installation (if both commands failed).

2.4 Managing ARP

The arp program manages entries in the system's Address Resolution Protocol (ARP) tables.We may recall that ARP provides the link between the IP address and the underlying physical address. Using arp (or its equivalent in other operating systems), the administrator can create, modify, or delete entries in the ARP table. Typically, this has to be performed whenever a machine's network address changes (either because of a change in the network hardware or because of a physical move).

The arp program differs considerably between implementations and is seldom used by users, so examples of its use are left to the operating system's configuration and administration documentation.

2.5 Using ifconfig

The ifconfig program, or one like it, enables an administrator to activate and deactivate network interfaces, as well as to configure them. Access to the ifconfig program is generally restricted to a superuser or network administrator. Changes to the configuration can usually be made only before the system is fully operational (such as in single-user mode on a UNIX system). When issued, ifconfig essentially instructs the network layer of the kernel to work with the specified network interface by assigning an IP address, then issuing a command to make the interface active on the system. Only when the interface is active can the operating system kernel send and receive data through the interface.

The ifconfig program enables a network administrator to perform several useful functions on most operating systems:

Activate or deactivate an interface

Activate or deactivate ARP on an interface

Activate or deactivate debugging mode on an interface

Assign a broadcast address

Assign a subnetwork mask

Assign a routing method

Examining all the options available to ifconfig would require several dozen pages. Because this material is rarely used and differs with each implementation, administrators are referred to their operating system documentation. As an example, the Linux version of the ifconfig command uses this general format:

```
ifconfig interface_type IP_Address
```

I nterface_type is the interface's device driver name (such as lo for loopback, ppp for PPP, and eth for Ethernet), and IP_Address is the IP address used by that interface. When used with only the name of an interface, ifconfig usually returns information about the current state of the interface, as shown in the following example. In this example, a query of both an Ethernet card (called ec0) and the loopback driver (called lo0) is performed. The status flags of the interface are followed by the Internet address, the broadcast address, and optionally a network mask, which defines the Internet address used for address comparison when routing.

```
tpci_sco1-12> ifconfig ec0
ec0: flags=807<UP,BROADCAST,DEBUG,ARP>
   inet 146.8.12.15 netmask ffffff00 broadcast
146.8.12.15
tpci_sco1-13> ifconfig lo0
lo0: flags=49<UP,LOOPBACK,RUNNING>
   inet 127.0.0.1 netmask ff000000
```

The preceding example shows that the Ethernet connection ec0 is active (UP), able to transmit broadcasts (BROADCAST), and is in debugging mode (DEBUG). Also, the ARP protocol is active (ARP). You may recall that a broadcast message is sent to all machines on the local network by setting the host ID address to all 1s.

Once the ifconfig command has been run and an interface is active, many operating systems require the route command to be issued to add or remove routes in the kernel's routing table. This is needed to enable the local machine to find other machines. The general format of the route command on a UNIX or Linux system is this:

```
route add|del IP_Address
```

Either add or del is specified to add or remove the route from the kernel's routing table, and IP_Address is the remote route being affected.

The current contents of the kernel's routing table can be displayed on some systems by entering the command route by itself on the command line. For example, on a Linux system that is set up only with the loopback driver, you see an output like this:

```
$ route
Kernel Routing Table
Destination   Gateway  Genmask  Flags  MSS  Window  Use Iface
loopback       *       255.0.0.0  U   1936  0       16  lo
```

The important columns are the destination name, which shows the name of the configured target (in this case only loopback), the mask to be used (Genmask), and the interface (Iface, in this case /dev/lo). You can force route to display the IP addresses instead of symbolic names by using the -n option:

```
$ route -n
Kernel Routing Table
Destination   Gateway  Genmask  Flags  MSS  Window  Use Iface
127.0.0.1      *       255.0.0.0  U   1936  0       16  lo
```

Not all UNIX and Linux versions show this type of output from the route command.

The use of the ifconfig and route programs can be shown in the setup of a Slackware Linux system's Ethernet connection. To make the Ethernet interface active, the ifconfig command is issued with the Ethernet device name (eth0 on a Slackware Linux system) and the local IP address. For example, the command

```
ifconfig eth0 147.123.20.1
```

sets up the local machine with the IP Address 147.123.20.1. The interface is the Ethernet device /dev/eth0. The interface can then be checked with the ifconfig command using the interface name:

```
$ ifconfig eth0
eth0    Link encap 10Mps: Ethernet Hwaddr
    inet addr 147.123.20.1 Bcast 147.123.1.255 Mask 255.255.255.0
    UP BROADCAST RUNNING  MTU 1500 Metric 1
    RX packets:0 errors:0 dropped:0 overruns:0
    TX packets:0 errors:0 dropped:0 overruns:0
```

You may notice in the output that the broadcast address was set based on the local machine's IP address. This is used by TCP/IP to access all machines on the local area network at once. The Message Transfer Unit (MTU) size is usually set to the maximum value of 1500 (for Ethernet networks).

Next, an entry is added to the kernel routing tables to let the kernel know about the local machine's network address. The IP address that is used with the route command is not your local machine's IP address, but that of the network as a whole without the local identifier. To set the entire local are network at once, the -net option of the route command is used. In the case of the IP addresses shown earlier, the command would be this:

```
route add -net 147.123.20.0
```

This adds all the machines on the network identified by the network address 147.123.20 to the kernel's list of accessible machines. An alternative method is to use the /etc/networks file. Once the route has been added to the kernel routing tables, it can be tested with the ping command.

2.6 The inetd Daemon

The inetd program is a holdover from the early days of TCP/IP UNIX development. When a UNIX machine was started, it would activate TCP/IP and immediately accept connections at its ports, spawning a process for each. This could result in many identical processes, one for each available port.

To control the processes better, the inetd program was developed to handle the port connections itself, offloading that task from the server. The primary difference is that inetd creates a process for each connection that is established, whereas the server creates a process for each port (which leads to many unused processes).

On many systems, some of the test programs and status information utilities are run through inetd. Typically, services like echo, discard, and time use inetd.

The inetd program uses a configuration file usually called /etc/inetd.cfg, /etc/inetd.conf, or /etc/inetd.cf on UNIX systems. An extract of a sample /etc/inetd.cfg file is shown in the following code:

```
#     @(#)inetd.conf    5.2 Lachman System V STREAMS TCP  source
#     System V STREAMS TCP - Release 4.0
ftp      stream   tcp   nowait   NOLUID   /etc/ftpd      ftpd
telnet   stream   tcp   nowait   NOLUID   /etc/telnetd   telnetd
shell    stream   tcp   nowait   NOLUID   /etc/rshd      rshd
login    stream   tcp   nowait   NOLUID   /etc/rlogind   rlogind
exec     stream   tcp   nowait   NOLUID   /etc/rexecd    rexecd
finger   stream   tcp   nowait   nouser   /etc/fingerd   fingerd
comsat   dgram    udp   wait     root     /etc/comsat    comsat
ntalk    dgram    udp   wait     root     /etc/talkd     talkd
echo     stream   tcp   nowait   root     internal
discard  stream   tcp   nowait   root     internal
chargen  stream   tcp   nowait   root     internal
daytime  stream   tcp   nowait   root     internal
time     stream   tcp   nowait   root     internal
echo     dgram    udp   wait     root     internal
discard  dgram    udp   wait     root     internal
chargen  dgram    udp   wait     root     internal
daytime  dgram    udp   wait     root     internal
time     dgram    udp   wait     root     internal
```

The columns show the service name (which corresponds to an entry in the services file, such as /etc/services), the socket type (stream, raw, or datagram), the protocol name, whether inetd can accept further connections at the same port immediately (nowait) or must wait for the server to finish (wait), the login that owns the service, the server program name, and any optional parameters needed for the server program.

The configuration file is read when the server is booted and every time a hang-up signal is received from an application. This enables dynamic changes to the file, because any modifications would be read and register on the next file read.

2.7 The netstat Command

The netstat program or a similar utility provides comprehensive information about the local system and its TCP/IP implementation. This is the program most commonly used by administrators to quickly diagnose a problem with TCP/IP. The actual information and its format supplied by the netstat utility differs with the operating system implementation, but it usually supplies the following important summaries:

Communications end points
Network interface statistics
Information on the data buffers
Routing table information
Protocol statistics

On some systems, information about the interprocess communications and other protocol stacks might be appended. The information to be displayed can usually be toggled with a command-line option. The output from a typical UNIX installation that uses the netstat command is shown in the next few sections, which discuss netstat and its output in more detail. The output and meaning might be different with other operating systems, but the general purpose of the diagnostic tool remains the same.

2.7.1 Communications End Points

The netstat command with no options provides information on all active communications end points. To display all end points (active and passive), netstat uses the -a option.

The output is formatted into columns showing the protocol (Proto), the amount of data in the receive and send queues (Recv-Q and Send-Q), the local and remote addresses, and the current state of the connection. A truncated sample output is shown here:

```
$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address         Foreign Address       (state)
ip        0      0  *.*                  *.*
tcp       0   2124  tpci.login           merlin.1034           ESTABL.
tcp       0      0  tpci.1034            prudie.login          ESTABL.
tcp   11212      0  tpci.1035            treijs.1036           ESTABL.
tcp       0      0  tpci.1021            reboc.1024            TIME_WAIT
tcp       0      0  *.1028               *.*                   LISTEN
tcp       0      0  *.*                  *.*                   CLOSED
tcp       0      0  *.6000               *.*                   LISTEN
tcp       0      0  *.listen             *.*                   LISTEN
tcp       0      0  *.1024               *.*                   LISTEN
tcp       0      0  *.sunrpc             *.*                   LISTEN
tcp       0      0  *.smtp               *.*                   LISTEN
tcp       0      0  *.time               *.*                   LISTEN
tcp       0      0  *.echo               *.*                   LISTEN
tcp       0      0  *.finger             *.*                   LISTEN
tcp       0      0  *.exec               *.*                   LISTEN
tcp       0      0  *.telnet             *.*                   LISTEN
tcp       0      0  *.ftp                *.*                   LISTEN
tcp       0      0  *.*                  *.*                   CLOSED
udp       0      0  *.60000              *.*
udp       0      0  *.177                *.*
udp       0      0  *.1039               *.*
udp       0      0  *.1038               *.*
udp    0    0 localhost.1036        localhost.syslog
udp    0    0 *.1034             *.*
udp    0    0 *.*                *.*
udp    0    0 *.1027             *.*
udp    0    0 *.1026             *.*
udp    0    0 *.sunrpc           *.*
udp    0    0 *.1025             *.*
udp    0    0 *.time             *.*
udp    0    0 *.daytime          *.*
udp    0    0 *.chargen          *.*
udp    0    0 *.route            *.*
udp    0    0 *.*                *.*
```

In the preceding example, there are three active TCP connections, as identified by the state ESTABL. One has data being sent (as shown in the Send-Q column), and another has incoming data in the queue. The network names and port numbers of the connection ends are shown whenever possible. An asterisk (*) means there is no end point associated with that address yet.

One connection is waiting to be hung up, identified by TIME_WAIT in the state column. After 30 seconds, these sessions are terminated and the connection freed. Any row with LISTEN as the state has no connection at the moment, and is waiting. There is no state column for UDP sessions because they do not have an end-to-end connection, A CLOSED entry in the output shows that the connection is closed but hasn't switched over to LISTEN yet.

2.7.2 Network Interface Statistics

The behavior of the network interface (such as the network interface card) can be determined with the -i option to the netstat command. This information quickly shows an administrator whether there are major problems with the network connection.

The netstat -i command displays the name of the interface, the maximum number of characters a packet can contain (Mtu), the network and host addresses or names, the number of input packets (Ipkts), input errors (Ierrs), output packets (Opkts), output errors (Oerrs), and number of collisions (Collis) experienced in the current sampling session. The collisions column has relevance only for a networking system that enables packet collisions, such as Ethernet. A sample output from a netstat -i command is shown here:

```
$ netstat -i
Name   Mtu   Network     Address       Ipkts   Ierrs Opkts   Oerrs Collis
ec0    1500  tpci        merlin        34      0     125     0     0
lan0   1497  47.80       tpci_hpws4    11625   0     11625   0     0
lo0    8232  loopback    localhost     206     0     206     0     0
```

An administrator can obtain more specific information about one interface by using the -I option with a device name and a time interval, specified in seconds, such as netstat -I ec0 30 to obtain specific information about the behavior of the ec0 (Ethernet) interface over the last 30 seconds.

2.7.3 Data Buffers

Information about the data buffers can be obtained with the netstat command's -m option. Monitoring the behavior of the buffers is important, because they directly impact the performance of TCP/IP. The output of the netstat -m command differs depending on the version of UNIX in use, reflecting the different implementations of the TCP/IP code.

The netstat -m command output from a System V-based UNIX version is shown in the following code example. Entries are provided for the streamhead, queue, message descriptor table (mblks), data descriptor table (dblks), and the different classes of data descriptor tables. The columns show the number of blocks configured (config) and currently allocated (alloc), the number of columns free (free), the total number of blocks in use (total), the maximum number of blocks that were in use at one time (max), and the number of times a block was not available (fail).

```
$ netstat -m
streams allocation:
               config  alloc  free  total  max  fail
streams          292     79    213   233    80    0
queues          1424    362   1062   516   368    0
```

```
mblks               5067    196   4871   3957    206     0
dblks               4054    196   3858   3957    206     0
class 0,    4 bytes   652     50    602    489     53     0
class 1,   16 bytes   652      2    650    408      4     0
class 2,   64 bytes   768      6    762   2720     14     0
class 3,  128 bytes   872    105    767    226    107     0
class 4,  256 bytes   548     21    527     36     22     0
class 5,  512 bytes   324     12    312     32     13     0
class 6, 1024 bytes   107      0    107      1      1     0
class 7, 2048 bytes    90      0     90      7      1     0
class 8, 4096 bytes    41      0     41     38      1     0
total configured streams memory: 1166.73KB
streams memory in use: 44.78KB
maximum streams memory used: 58.57KB
```

For the administrator, the failure column is important. It should always show 0s. If a larger number appears, that resource has been overtaxed and the number of blocks assigned to that resource should be increased (followed by a kernel rebuild and a reboot of the system to effect the changes).

2.7.4 Routing Table Information

Routing tables are continually updated to reflect connections to other machines. To obtain information about the routing tables, the netstat -r and -rs options are used. (The latter generates statistics about the routing tables.)

The output from netstat -r and netstat -rs commands are shown in the following code example. The columns show the destination machine, the address of the gateway to be used, a flag to show whether the route is active (U) and whether it leads to a gateway or a machine (H for host), a reference counter (Refs) that specifies how many active connections can use that route simultaneously, the number of packets that have been sent over the route (Use), and the interface name.

```
$ netstat -r
Routing tables
```

```
Destination     Gateway         Flags   Refs    Use  Interface
localhost       localhost        UH       4      10  lo0
merlin          localhost        UH       2       2  ec0
treijs          hoytgate         UG       0       0  ec0
47.80           bcarh736         U       12   21029  lan0
tpci sco4-57> netstat -rs
routing:
          0 bad routing redirects
          0 dynamically created routes
          0 new gateways found unreachable
          2 destinations found unreachable
        122 uses of a wildcard route
          0 routes marked doutbful
          0 routes cleared of being doubtful
          0 routes deleted
```

2.7.5 Protocol Statistics

Statistics about the overall behavior of network protocols can be obtained with the netstat -s command. This usually provides summaries for IP, ICMP, TCP, and UDP. The output from this command is useful for determining where an error in a received packet was located, which then leads the user to isolate whether that error was caused by a software or network problem.

Issuing the netstat -s command provides a verbose output. A sample output is shown in the following code. The entries are self-explanatory.

```
tpci_sco4-67> netstat -s
ip:
    183309 total packets received
    0 bad header checksums
    0 with size smaller than minimum
    0 with data size < data length
    0 with header length < data size
    0 with data length < header length
    0 with unknown protocol
    13477 fragments received
    0 fragments dropped (dup or out of space)
    0 fragments dropped after timeout
    0 packets reassembled
    0 packets forwarded
    0 packets not forwardable
    75 no routes
    0 redirects sent
    0 system errors during input
    309 packets delivered
    309 total packets sent
    0 system errors during output
    0 packets fragmented
    0 packets not fragmentable
    0 fragments created
icmp:
    1768 calls to icmp_error
    0 errors not generated because old message was icmp
    Output histogram:
        destination unreachable: 136
    0 messages with bad code fields
    0 messages < minimum length
    0 bad checksums
    0 messages with bad length
    Input histogram:
        destination unreachable: 68
    0 message responses generated
    68 messages received
    68 messages sent
    0 system errors during output
tcp:
   9019 packets sent
        6464 data packets (1137192 bytes)
    4 data packets (4218 bytes) retransmitted
    1670 ack-only packets (918 delayed)
    0 URG only packets
    0 window probe packets
    163 window update packets
    718 control packets
    24 resets
    9693 packets received
    4927 acks (for 74637 bytes)
    37 duplicate acks
    0 acks for unsent data
    5333 packets (1405271 bytes) received in-sequence
    23 completely duplicate packets (28534 bytes)
    0 packets with some dup. data (0 bytes duped)
    38 out-of-order packets (5876 bytes)
    0 packets (0 bytes) of data after window
    0 window probes
    134 window update packets
    0 packets received after close
    0 discarded for bad checksums
    0 discarded for bad header offset fields
    0 discarded because packet too short
    0 system errors encountered during processing
224 connection requests
130 connection accepts
687 connections established (including accepts)
655 connections closed (including 0 drops)
24 embryonic connections dropped
0 failed connect and accept requests
0 resets received while established
5519 segments updated rtt (of 5624 attempts)
5 retransmit timeouts
0 connections dropped by rexmit timeout
0 persist timeouts
```

```
    0 keepalive timeouts
    0 keepalive probes sent
    0 connections dropped by keepalive
    0 connections lingered
        0 linger timers expired
        0 linger timers cancelled
        0 linger timers aborted by signal
udp:
    0 incomplete headers
    0 bad data length fields
    0 bad checksums
    68 bad ports
    125 input packets delivered
    0 system errors during input
    268 packets sent
```

2.8 The ping Utility

The ping (Packet Internet Groper) utility is used to query another system to ensure that a connection is still active. (You may recall the ruptime utility from yesterday, which also does this. However, ruptime waits five minutes before trying the remote, and you may want to know right away if the connection is active.) The ping command is available on most operating systems that implement TCP/IP.

The ping program operates by sending out an Internet Control Message Protocol (ICMP) echo request. If the destination machine's IP software receives the ICMP request, it issues an echo reply immediately. The sending machine continues to send an echo request until the ping program is terminated with a break sequence (Ctrl+C or the Delete key in UNIX). After termination, ping displays a set of statistics. A sample ping session is shown here:

```
$ ping merlin
PING merlin: 64 data bytes
64 bytes from 142.12.130.12: icmp_seq=0.  time=20.  ms
64 bytes from 142.12.130.12: icmp_seq=1.  time=10.  ms
```

```
64 bytes from 142.12.130.12: icmp_seq=2.  time=10.  ms
64 bytes from 142.12.130.12: icmp_seq=3.  time=20.  ms
64 bytes from 142.12.130.12: icmp_seq=4.  time=10.  ms
64 bytes from 142.12.130.12: icmp_seq=5.  time=10.  ms
64 bytes from 142.12.130.12: icmp_seq=6.  time=10.  ms
--- merling PING Statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms) min/avg/max = 10/12/20
```

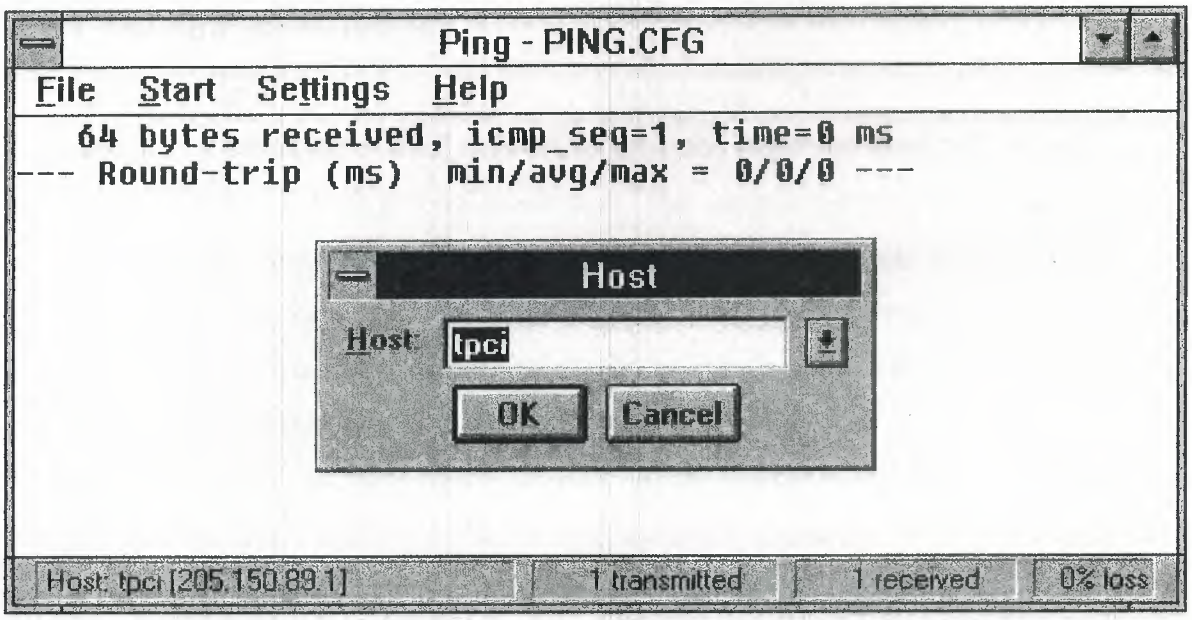
An alternate method to invoke ping is to provide the number of times you want it to query the remote. Also, you could provide a packet length as a test. The following example instructs ping to use 256 data byte packets and try five times. Using ping to send large packets is one method of determining the network's behavior with large packet sizes, especially when fragmentation must occur. The ping program is also useful for monitoring response times of the network, by observing the reply time on packets sent as the network load (or the machine load) changes. This information can be very useful in optimization of TCP/IP.

```
$ ping merlin 256 5
PING merlin: 256 data bytes
256 bytes from 142.12.130.12: icmp_seq=0.  time=20.  ms
256 bytes from 142.12.130.12: icmp_seq=1.  time=10.  ms
256 bytes from 142.12.130.12: icmp_seq=2.  time=10.  ms
256 bytes from 142.12.130.12: icmp_seq=3.  time=20.  ms
256 bytes from 142.12.130.12: icmp_seq=4.  time=10.  ms
--- merling PING Statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms) min/avg/max = 10/13/20
```

Some older implementations of ping simply reply with a message that the system at the other end is active. (The message is of the form X is alive.) To obtain the verbose messages shown previously, the -s option must be used.

The ping program is useful for diagnostics because it provides four important pieces of information: whether the TCP/IP software is functioning correctly; whether a local network device can be addressed (validating its address); whether a remote machine can be accessed (again validating the address and testing the routing); and verifying the software on the remote machine.

Most non-UNIX TCP/IP implementations provide ping utilities as part of their suite. For example, Figure 2.4 shows the NetManage ChameleonNFS ping utility. The Chameleon ping sends only a single ICMP packet instead of a continuous stream, but is useful for verifying that a remote machine is responding.



```
   64 bytes received, icmp_seq=1, time=0 ms
--- Round-trip (ms)  min/avg/max = 0/0/0 ---
```

Host: tpci

Host: tpci [205.150.89.1] | 1 transmitted | 1 received | 0% loss

Figure 2.4 Net Manage Chameleon NFS ping utility

2.9 Tracing a Connection

There is a tracing option built into TCP/IP. When simpler methods have failed, this option can be used to trace a problem. To activate the trace, a system call is sent to the end point that turns on a flag. Most TCP/IP implementations enable the tracing option to be turned on from the command line using the -d (debug) option. When tracing is turned on, all activities are echoed to a buffer or to the screen, depending on the system configuration.

The output from the TCP/IP tracing option is examined using the program trpt (trace report). A specific connection can be specified, or all behavior passing through TCP/IP can be displayed. The output from trpt is verbose and of little interest to most users.

3. TCP/IP BASED SERVER AND CLIENT

3.1 Setting Up a Sample TCP/IP Network: Server

How to set up a TCP/IP Server on Windows NT machine, Every server are connected to the sample network, and any of them can be accessed by a client machine or other server.

3.1.1 The Sample Network

I designed a dedicated TCP/IP network to show the steps you must follow to set up, configure, and test a TCP/IP implementation. The sample network relies on several servers, although many networks have only one. Also, I use several different types of servers to show you how they can be configured, whereas most real networks are not this diverse. All the machines are connected over an Ethernet network. In all, the sample network has four servers and three clients.

Each of the seven machines on the network has its own name and IP address. For this sample network, the IP address mask has been randomly chosen as 147.120. The names of the machines have been chosen from my pets, although any unique name would do, of course.

The physical setup of the network is undertaken first. It involves installing a network interface card in each machine (except the SPARCstation, which has the network card as part of the motherboard). On each system you must ensure that any jumpers for interrupt vectors and memory I/O addresses do not conflict with any other card on that system. (Some of the cards are software programmable; some are set by jumpers or DIP switches.) All the boards used in this system are from different manufacturers to show the independent nature of the TCP/IP network.

Cable must be run between all the machines, connecting the network interface cards together. In the case of Ethernet, the cables must be properly terminated. The sample network uses thin Ethernet, which closely resembles television coaxial cable. BNC Thin Ethernet connectors resemble a T, with cables attached to both ends of the T and the stem connected to the network card. Two of the machines form the ends of the cable and require a terminating resistor as part of their T. The SPARCstation normally uses an

RJ45 connector (which looks like a wide telephone connector, so I used a transceiver to convert it to BNC).

To test the physical network, it is easiest to wait until a couple of machines have had their basic software configuration completed. All the machines on the network do not have to be active, as long as the network cable is contiguous from end to end and each BNC connector is attached to a network card to provide electrical termination. If problems are found when the network is tested, the physical network is the first item to check. Some network monitoring devices can supply integrity information prior to installing the network, but these devices are not usually available to system administrators who are just beginning their installation, or who have a small number of machines to maintain (primarily because the network testers tend to be expensive).

3.1.2 Configuring TCP/IP Software

Most operating systems and TCP/IP software packages provide several utilities, including menu-driven scripts that help automate the installation process of the TCP/IP applications. Some operating systems still require manual configuration of several files using a text editor. To configure TCP/IP software properly, you must know several pieces of information before you start. The necessary information you need for each machine on the network follows:

- Domain name: The name the entire network will use.
- System name: The unique name of each local machine.
- IP address: The full address of each machine.
- Driver type: Each interface to the network must be associated with a device driver, instructing the operating system how to talk to the device.
- Broadcast address: The address used for network-wide broadcasts.
- Netmask: The network mask that uniquely identifies the local network.
- Hardware network card configuration information: The interrupt vector and memory address of the network card.

The system domain name is necessary if the network is to be connected to other machines outside the local network. Domain names can be invented by the system administrator. If, however, the network is to interface with Internet or one of its service providers, the domain name should be approved by the Internet Network Information Center (InterNIC). Creating and registering a new domain is as simple as filling out a form (and recently, paying a small administration fee). Domain names usually reflect the company name, with the extension identifying the type of organization. The sample network uses the name tpci.com.

The machine name is used for symbolic naming of a machine instead of forcing the full IP address to be specified. The system name must be unique on the local network. Other networks might have machines with the same name, but their network masks are different, so there is no possible confusion during packet routing. In most cases, system names are composed of eight characters (or less) and are usually all lowercase characters. The system name can be a mix of characters and numbers. Larger organizations tend to number their machines, and small companies give their machines more familiar names.

The device driver instructs the operating system how to communicate with the network interface (usually either a network card or a serial port). Each interface has its own specific device driver. Most operating systems have device drivers included in their distribution software, although some require software supplied with the network card. Generic drivers are available for most network cards on bulletin board systems.

The network card configuration must be known in order to install the device driver properly. Network cards usually have several configuration settings, depending on the system for which they are designed. For the PC-based machines in the sample network, each card must have a unique interrupt vector (called an IRQ) and a unique I/O memory address. IRQ and address settings on many of the newer network boards are software-configurable, making the installation and configuration much easier.

Most network cards come with default settings that might conflict with other cards in the system. Users must carefully check for conflicts, resorting to a diagnostic program if available. UNIX users have several utilities available, depending on the operating system. SCO UNIX and most System V Release 4 operating systems have the utility hwconfig, which shows the current hardware configuration. The following example shows the hwconfig output and the output from the command with the -h option to provide long formatting with headers (making it is easier to read):

```
$ hwconfig
name=fpu vec=13 dma=- type=80387
name=serial base=0x3F8 offset=0x7 vec=4 dma=- unit=0 type=Standard nports=1
name=serial base=0x2F8 offset=0x7 vec=3 dma=- unit=1 type=Standard nports=1
```

```
name=floppy base=0x3F2 offset=0x5 vec=6 dma=2 unit=0 type=96ds15
name=floppy vec=- dma=- unit=1 type=135ds18
name=console vec=- dma=- unit=vga type=0 12 screens=68k
name=adapter base=0x2C00 offset=0xFF vec=11 dma=- type=arad ha=0 id=7 fts=st
name=nat base=0x300 offset=0x20 vec=7 dma=- type=NE2000 addr=00:00:6e:24:1e:3e
name=tape vec=- dma=- type=S ha=0 id=4 lun=0 ht=arad
name=disk vec=- dma=- type=S ha=0 id=0 lun=0 ht=arad fts=stdb
name=Sdsk vec=- dma=- cyls=1002 hds=64 secs=32
$
$ hwconfig -h
device      address    vec dma  comment
fpu            -       13  -  type=80387
serial    0x3f8-0x3ff   4  -  unit=0 type=Standard nports=1
serial    0x2f8-0x2ff   3  -  unit=1 type=Standard nports=1
floppy    0x3f2-0x3f7   6  2  unit=0 type=96ds15
floppy         -        -  -  unit=1 type=135ds18
console        -        -  -  unit=vga type=0 12 screens=68k
adapter   0x2c00-0x2cff 11 -  type=arad ha=0 id=7 fts=st
nat       0x300-0x320   7  -  type=NE2000 addr=00:00:6e:24:1e:3e
tape           -        -  -  type=S ha=0 id=4 lun=0 ht=arad
disk           -        -  -  type=S ha=0 id=0 lun=0 ht=arad fts=stdb
Sdsk           -        -  -  cyls=1002 hds=64 secs=32
```

This output is from the SCO UNIX servers set up for the sample network. It has the network Ethernet card already configured as device nat, which uses IRQ 7 (shown under the vec or interrupt vector column). The nat line also shows the memory address as 300–320 (hexadecimal) and the device driver as NE2000 (a Novell NetWare-compatible driver). The address and vec columns show no conflicts between the settings used for the Ethernet card and other devices on the system. (The adapter entry is for a high-speed SCSI-2 card, which controls both the tape and the Sdsk device, the primary SCSI hard drive. All other entries should be self-explanatory.)

DOS users can use the Microsoft Diagnostic utility, MSD.EXE, or one of several third-party tools such as Central Point PC Tools or The Norton Utilities to display IRQ vectors and memory addresses in use by the system. Some software even indicates which vectors and addresses are available for use.

There is no need to have the same IRQ and memory address for each card on the network, because the network itself doesn't care about these settings. The IRQ and memory addresses are required for the machine to communicate with the network interface card only. The sample network used a different IRQ and memory address for each machine.

IRQ and memory addresses are usually set on the network interface card itself using either jumpers on pins or a DIP-switch block. The documentation accompanying the card should provide all the information necessary for setting these values. Some recently introduced network interface cards can be configured through software, enabling the settings to be changed without removing the card from the system. This can be very handy when a user is unsure of the best settings for the card.

The IP address is a 32-bit number that must be unique for each machine. If the network is to be connected to the Internet, the IP address must be assigned by the NIC (it is usually given to you when you register your domain name). Even if no access to the Internet is expected, arbitrarily assigning an IP address can cause problems when messages are passed with other networks. If the network is not connected to the outside world, a system administrator can ignore the NIC's numbering system and adopt any IP address. It is worthwhile, however, to consider future expansion and connection to other networks.

As you might recall, the NIC has four classes of IP addresses in use depending on the size of the network. Each class has some addresses that are restricted. These are shown in Table 3.1. Most networks are Class B, although a few large corporations require Class A networks.

Table 3.1. The NIC IP address classes.

Class	Network Mask Bytes	Number of Hosts per Network	Valid Addresses
A	1	16,777,216	1.0.0.1 to 126.255.255.254
B	2	65,534	128.0.0.1 to

			191.255.255.254
C	3	254	224.0.0.0 to 255.255.255.254
D	reserved		

The network mask is the IP address stripped of its network identifiers, leaving only the local machine address. For a Class A network, this strips one byte, whereas a Class B network strips two bytes (leaving two). The small Class C network strips three bytes as the network mask, leaving one byte to identify the local machine (hence the limit of 254 machines on the network). The sample network is configured as a Class B machine with the randomly chosen IP address network mask of 147.120 (not NIC-assigned).

The broadcast address identifies packets that are to be sent to all machines on the local network. Because a network card usually ignores any incoming packets that don't have its specific IP address in them, a special broadcast address can be set that the card can intercept in addition to locally destined messages. The broadcast address has the host portion (the local machine identifiers) set to either all 0s or all 1s, depending on the convention followed. For convenience, the broadcast address's network mask is usually the same as the local network mask.

The steps followed for configuring TCP/IP are straightforward, generally following the information required for each machine. The configuration steps are as follows:

- Link drivers: TCP/IP must be linked to the operating system's kernel or loaded during the boot stage to enable TCP/IP.
- Add host information: Provide a list of all machines (hosts) on the network (used for name resolution).
- Establish routing tables: Provide the information for routing packets properly if name resolution isn't sufficient.
- Set user access: Configure the system to enable access in and out of the network, as well as establishing permissions.
- Remote device access: Configure the system for access to remote printers, scanners, CD-ROM carousels, and other shared network devices.
- Configure the name domain server: If using a distributed address lookup system such as Berkeley Internet Name Domain Server (BIND) or NIS, complete the

name server files. (This step is necessary only if you are using BIND or a similar service.)

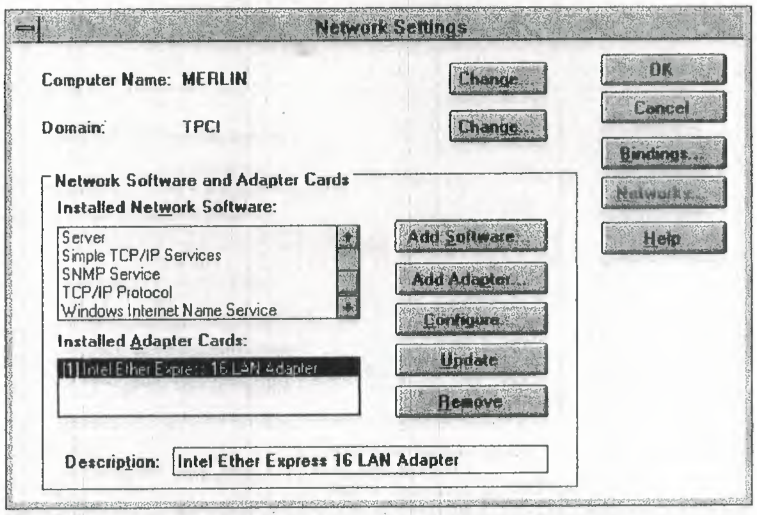
- Tune system for performance: Because a system running TCP/IP has different behavior than one without TCP/IP, some system tuning is usually required.
- Configure NFS: If the Network File System (NFS) is to be used, configure both the file system and the user access.
- Anonymous FTP: If the system is to enable anonymous FTP access, configure the system and public directories for this service.

You will use these steps (not necessarily in the sequence given) as the individual machines on the network are configured. The processes are different with each operating system, but the overall approach remains the same.

3.1.3 Configuring Windows NT Server

Windows NT is available in both server and workstation versions. I configure the server version for the sample network. I use Windows NT Server 3.51 on the sample system although Windows NT 4.0 performs in almost exactly the same way. Although TCP/IP is provided with Windows NT, it is not installed as the default network protocol. Instead, IPX/SPX and NetBEUI are installed as default protocols. To configure TCP/IP, you need to extract the TCP/IP software from the distribution media if it hasn't already been installed.

You can check for the presence of the TCP/IP software by opening the Network Settings window inside the Control Panel. This window is shown in Figure 3.1. The scroll list in the bottom left corner has a list of all installed components. If it does not include an entry such as TCP/IP Protocol, the TCP/IP software is not installed. To install the TCP/IP software, click the Add Software button on the Network Settings window.



Network Settings

Computer Name: MERLIN

Domain: TPCI

Network Software and Adapter Cards

Installed Network Software:

Server
Simple TCP/IP Services
SNMP Service
TCP/IP Protocol
Windows Internet Name Service

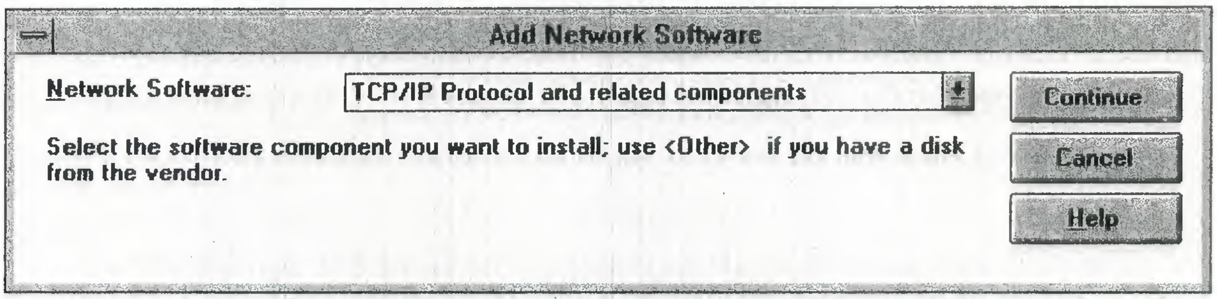
Installed Adapter Cards:

[1] Intel Ether Express 16 LAN Adapter

Description: Intel Ether Express 16 LAN Adapter

Figure 3.1 The Windows NT Network Settings shows all the components that are installed.

When you select Add Software, the system checks for all the installed and available components (which can take some time), then displays the windows shown in Figure 3.2. After selecting TCP/IP to be installed, you can select the specific TCP/IP components and any other TCP/IP services you want to install from the window shown in Figure 3.3.

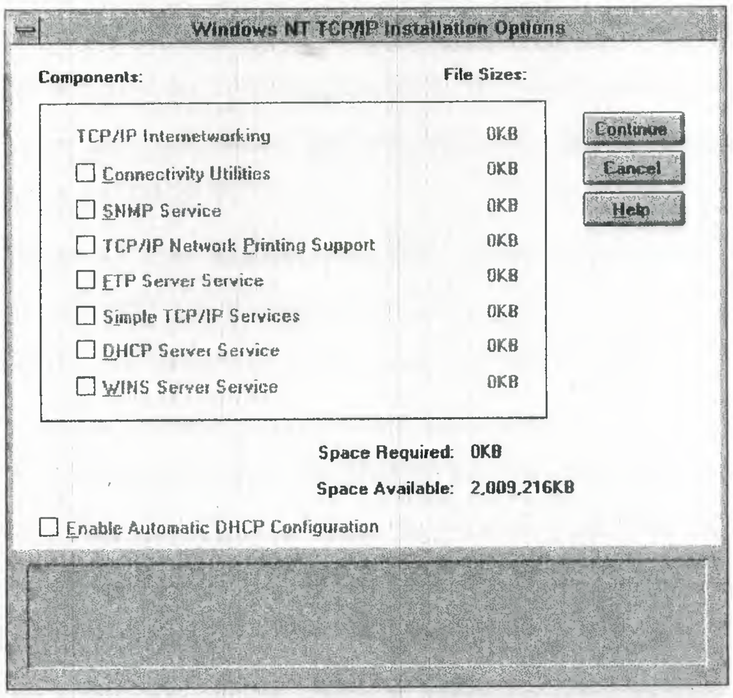


Add Network Software

Network Software: TCP/IP Protocol and related components

Select the software component you want to install; use <Other> if you have a disk from the vendor.

Figure 3.2 You can add the TCP/IP software to your Windows NT system through this window.


Windows NT TCP/IP Installation Options

Components: File Sizes:

TCP/IP Internetworking 0KB

☐ Connectivity Utilities 0KB

☐ SNMP Service 0KB

☐ TCP/IP Network Printing Support 0KB

☐ FTP Server Service 0KB

☐ Simple TCP/IP Services 0KB

☐ DHCP Server Service 0KB

☐ WINS Server Service 0KB

Space Required: 0KB

Space Available: 2,009,216KB

☐ Enable Automatic DHCP Configuration

Figure 3.3 Select the components of the Windows NT TCP/IP software that you want to install from this window.

The server version of Windows NT offers several TCP/IP configuration options and extra services. Those shown in Figure 3.3 include the following:

- TCP/IP Internetworking: These must be installed for TCP/IP to function. It includes the drivers for TCP, IP, UDP, and ARP, as well as several other protocols like ICMP. PPP and SLIP are also provided through this option.
- Connectivity Utilities: Utilities like finger, ping, telnet, and many others. These should be installed with all TCP/IP configurations.
- SNMP Service: The SNMP drivers used to enable the server or workstation to be administered remotely. This option should be used if your Windows NT machine is to be managed by a remote UNIX workstation. The SNMP Service is also required if you want to run the Performance Monitor and obtain TCP/IP behavior statistics.
- TCP/IP Network Printing: Enables network printers (those attached directly to the network cables instead of a PC) to be used. This option can also be used if you want to send all print requests on this machine to another machine for handling, such as a UNIX print server.
- FTP Server Service: If you want to use FTP to transfer files from Windows NT, this service must be loaded.

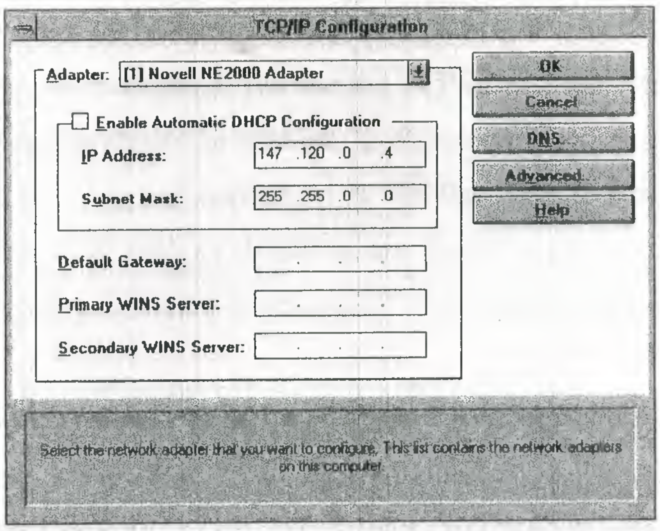
- Simple TCP/IP Services: Offers specialty services like Daytime, Echo, and Quote that are used by some applications. If you are using UNIX workstations on the same network, these services probably should be supported by the Windows NT machine.
- DHCP Server Service: Installs the DHCP server software. If you want to use DHCP on your network, you need a DHCP server.
- WINS Server: If WINS is to be used on your network, install the server software.

Clicking the OK button begins the installation process, with Windows NT prompting you for the distribution CD-ROM or disks as needed. After the TCP/IP software is installed, you have to reboot the machine and then the Network Settings window should show the TCP/IP protocols in place.

If you installed a network adapter when the Windows NT operating system software was loaded, the network adapter card should also show in the list of installed components in the Network Settings window. If you need to add a network adapter card to the system, it can be added through the Network Settings window, too. The Add Adapter button starts the installation routine, which prompts for the type of network adapter card, then the settings on the card for IRQ and memory address. After the network card has been configured, the drivers are loaded by Windows NT, then a system reboot makes the card available.

The Network Settings window lets you configure each component of the TCP/IP software installed on the Windows NT server. You can change the machine name and domain name from the Network Settings window by clicking the Change button next to those items at the top of the screen. Only an administrator can change the machine and domain names.

If you highlight TCP/IP Protocol in the Network Settings window, then click the Configure button, you see the TCP/IP Configuration window shown in Figure 3.4. This lets you provide the IP address of the local machine (assuming it is not assigned through the use of another service like DHCP or WINS). If you are using a DHCP or WINS server (other than the machine you are configuring now), the IP address of that server should be entered on this screen.



Adapter: [1] Novell NE2000 Adapter

Enable Automatic DHCP Configuration [ ]

IP Address: 147 .120 .0 .4

Subnet Mask: 255 .255 .0 .0

Default Gateway: . . .

Primary WINS Server: . . .

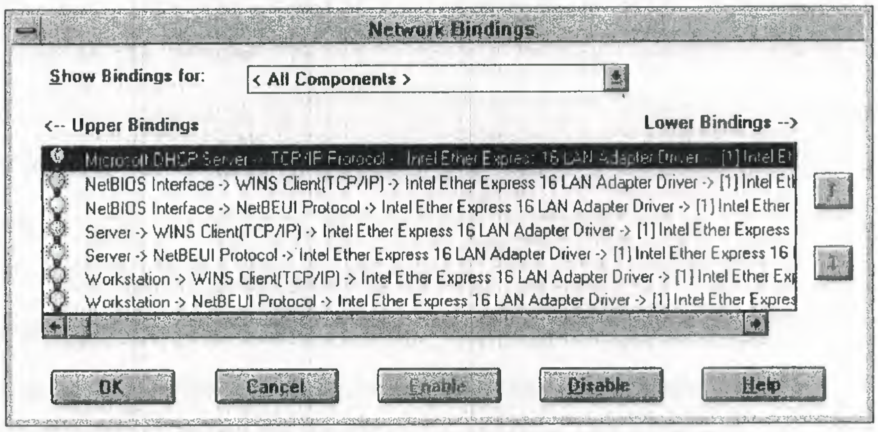
Secondary WINS Server: . . .

Select the network adapter that you want to configure. This list contains the network adapters on this computer.

Figure 3.4 The IP address of the local machine is entered in this window.

If you are using DNS on your network, select the DNS button in the TCP/IP Configuration window. This displays the DNS Configuration window. This window lets you specify the hostname and domain name of the DNS server as well as any specifics about the DNS server search order. If you are not using DNS, you can leave this window as it is. Because you are not setting up a DNS server at the moment, you can leave this window alone. Finally, the Advanced button on the TCP/IP Configuration window lets you select subnet masks and gateway IP addresses, if necessary.

From the Network Settings window, you should check the network bindings to make sure TCP/IP is used for communications over the local area network. Select the Bindings button on the Network Settings window to display the Network Bindings window, shown in Figure 3.5.



Show Bindings for: < All Components >

```
Microsoft DHCP Server -> TCP/IP Protocol -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Et
NetBIOS Interface -> WINS Client(TCP/IP) -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Et
NetBIOS Interface -> NetBEUI Protocol -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Ether
Server -> WINS Client(TCP/IP) -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Ether Express
Server -> NetBEUI Protocol -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Ether Express 16
Workstation -> WINS Client(TCP/IP) -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Ether Ex
Workstation -> NetBEUI Protocol -> Intel Ether Express 16 LAN Adapter Driver -> [1] Intel Ether Expres
```

Figure 3.5 The Network Bindings window shows all network bindings configured on the system.

If TCP/IP is properly configured, you see the TCP/IP protocol bound to the network adapter card. The binding should be enabled, as shown by a yellow lightbulb to the left of the binding name. If it is not enabled, click the Enable button at the bottom of the window. If other protocols, such as IPX/SPX, are bound to the same network card and enabled but not needed, you should disable them. Only leave the bindings that you need enabled.

After the configuration information has been verified, you should click Update or OK and allow Windows NT to complete the configuration for you. You might have to provide the source disks or CD-ROM if new software is necessary. After the configuration is complete, you need to reboot the machine to effect any changes.

To verify that the configuration is working properly, you should run the ping command and try pinging another machine on the network. The ping utility is DOS-based and can usually be found under WINNT35\SYSTEM32. Start a DOS session and issue the ping command, followed by a known IP address. If the remote is successfully pinged, your installation and configuration are working.

3.1.4 Testing the Server Configurations

Testing the TCP/IP configuration on any of the four configured servers is straightforward. Begin by using ping on each machine to ensure that the software is talking to the network hardware. Unfortunately, a successful ping of the local machine does not always mean the network is being accessed properly; it simply means the network software is processing the request. To test the network interface itself, ping the other machines on the network. In the following example, merlin is the local host and sinbad is a DOS machine running ftp Software's PC/TCP (which you see tomorrow):

```
$ ping merlin
PING localhost (147.120.0.1): 56 data bytes
64 bytes from localhost (147.120.0.1): icmp_seq=0 ttl=255 time=0 ms
64 bytes from localhost (147.120.0.1): icmp_seq=1 ttl=255 time=0 ms
64 bytes from localhost (147.120.0.1): icmp_seq=2 ttl=255 time=0 ms
64 bytes from localhost (147.120.0.1): icmp_seq=3 ttl=255 time=0 ms
64 bytes from localhost (147.120.0.1): icmp_seq=4 ttl=255 time=0 ms
```

```
--- localhost ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms
$ ping sinbad
PING sinbad (147.120.0.11): 56 data bytes
64 bytes from localhost (147.120.0.1): icmp_seq=0 ttl=255 time=20 ms
64 bytes from localhost (147.120.0.1): icmp_seq=1 ttl=255 time=20 ms
64 bytes from localhost (147.120.0.1): icmp_seq=2 ttl=255 time=50 ms
64 bytes from localhost (147.120.0.1): icmp_seq=3 ttl=255 time=30 ms
64 bytes from localhost (147.120.0.1): icmp_seq=4 ttl=255 time=40 ms
--- pepper ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 20/32/50 ms
```

The first test shows that the software is configured properly. The command to ping merlin resulted in a conversion within the /etc/hosts file to recognize the instruction as the localhost entry. After verifying the local connection, the remote machine is tried. The successful round-trip of the packets indicates that the remote is working properly, and that the network is functional. Of course, this works only if the remote machine has been loaded with TCP/IP software and is active.

If the localhost ping command failed, the software was probably configured incorrectly, or the hardware was not accessed properly. First, check the connectors on the network cards, because they have an annoying habit of working loose. Next, check the network configuration (IRQ, address, and type of adapter), followed by the configuration files, as shown earlier. If everything looks correct and the remote machine answers its own ping command properly, there is a problem with software compatibility. The netstat network status command is useful for monitoring the network's performance and detecting problems. TCP/IP system administrators frequently use the options -i, -m, and -s.

A common problem is the lack of enough STREAMS buffers, which causes a process to hang or a connection to terminate for no apparent reason. The size of the STREAMS buffer and its current status can be checked with the command netstat -m:

```
$ netstat -m
```

```
streams allocation:
                     config  alloc  free  total   max  fail
streams                 292     78   214    145    79     0
queues                 1424    360  1064    327   364     0
mblks                  5077    197  4880   3189   206     0
dblks                  4062    197  3865   3167   205     0
class 0,    4 bytes     652     51   601    357    53     0
class 1,   16 bytes     652      1   651    284     3     0
class 2,   64 bytes     768      8   760   2158    15     0
class 3,  128 bytes     872    104   768    237   106     0
class 4,  256 bytes     548     21   527     90    22     0
class 5,  512 bytes     324     12   312     13    13     0
class 6, 1024 bytes     107      0   107      1     1     0
class 7, 2048 bytes      98      0    98      1     1     0
class 8, 4096 bytes      41      0    41     26     1     0
total configured streams memory: 1183.09KB
streams memory in use: 44.66KB
maximum streams memory used: 58.28KB_
```

The number in the fail column should be 0 in each row; otherwise, there is a problem with the amount of buffer allocated. To change the number of STREAMS buffers allocated, kernel variables must be changed and the kernel relinked. As a general rule, if there are problems with the existing STREAMS buffer sizes, increase the number by 50 percent. If that doesn't solve the problem, increase by another 50 percent.

To fully test the TCP/IP system, use Telnet or FTP to log in and transfer files from machine to machine. Because these two utilities are the most common users of TCP/IP (unless NIS or NFS are active), they help show any problems with the port assignments, services provided, or name mapping.

3.2 Setting Up a Sample TCP/IP Network: Windows Client

The Windows NT server was configured using the built-in TCP/IP stack. Now we configure windows based client for the network. The clients communicate with the server through a TCP/IP stack loaded on each machine. Any of the operating systems you configured as servers can also act as clients on the sample network.

Windows includes TCP/IP client software as part of the distribution software package, but it is not configured when Windows is installed. This is because Windows installs NetWare's IPX/SPX network protocols as the default. Now you see how to change the default protocol to TCP/IP. Windows machines, several products are available to offer TCP/IP protocols. I have selected two of the most popular packages to configure on these systems. The Windows machine, running Microsoft Windows for Workgroups, is configured with NetManage's ChameleonNFS.

3.2.1 Windows-Based TCP/IP: NetManage's Chameleon

NetManage produces a line of TCP/IP-based software specifically for Windows, Windows, and Windows for Workgroups. These applications are designed to provide full access to TCP/IP utilities through the Windows environment. NetManage's line of products includes a basic TCP/IP stack (called Newt), as well as full TCP/IP application packages in several forms, all called Chameleon. The system is also available for Windows NT. You are installing Chameleon on a Windows for Workgroups machine on the sample network.

Chameleon uses the standard NDIS (Network Device Interface Specification) or the ODI (Open Data Link Interface) for communicating with the network interface card. This enables any card that uses either NDIS or ODI to be used with Chameleon.

Prior to installation of Chameleon, the same steps are performed as for the DOS-based TCP/IP package. The network interface card must be installed with suitable IRQ and memory address settings. If Chameleon is being added to an existing Windows for Workgroups system, the network card should already be installed and properly configured. The same information is required as for all TCP/IP installations: the host name, IP address, broadcast mask, subnetwork mask, and any information about gateways or routers that needs to be included.

The version of ChameleonNFS used for the sample network had its installation information slightly jumbled because of updates to both Chameleon and Windows for Workgroups. The information supplied today applies to Windows for Workgroups and ChameleonNFS version 4.0, although other versions should be similar.

3.2.2 Installing Chameleon

Chameleon can be installed over a fully functioning Windows or Windows for Workgroups system. If Windows for Workgroups is used, ensure that the network performs properly (if possible) when talking to other NetBEUI-compatible machines. In this case, that's not possible because the sample network uses only TCP/IP.

The installation procedure for Chameleon is simple. From the Program Manager's File menu, select Run, then execute the SETUP.EXE program from the first Chameleon disk. As with most Windows applications, this starts the installation program.

The changes made to the system files might cause problems, affecting Windows' capability to boot. Before installing the Chameleon software, make copies of the AUTOEXEC.BAT, CONFIG.SYS, PROTOCOL.INI, WIN.INI, and SYSTEM.INI files. If problems are encountered, these files can return the system to its original state. You should consider making a full system backup before any major changes to software, of course.

The Chameleon installation program requires a lengthy serial number and an activation key to ensure that there is only one such version on a network (this locks out multiple installations using the same serial number and activation key.) The installation script prompts for the distribution disks in order and copies all the necessary files.

Following the installation process, Chameleon builds the program group with the Chameleon applications included. The ChameleonNFS program group is shown in Figure 3.6. After creating the program group, Chameleon starts a customization screen that lets you specify your IP address, host name, network mask, and broadcast address. Save this information and then exit out of Windows to the DOS prompt to complete the check of the installation.

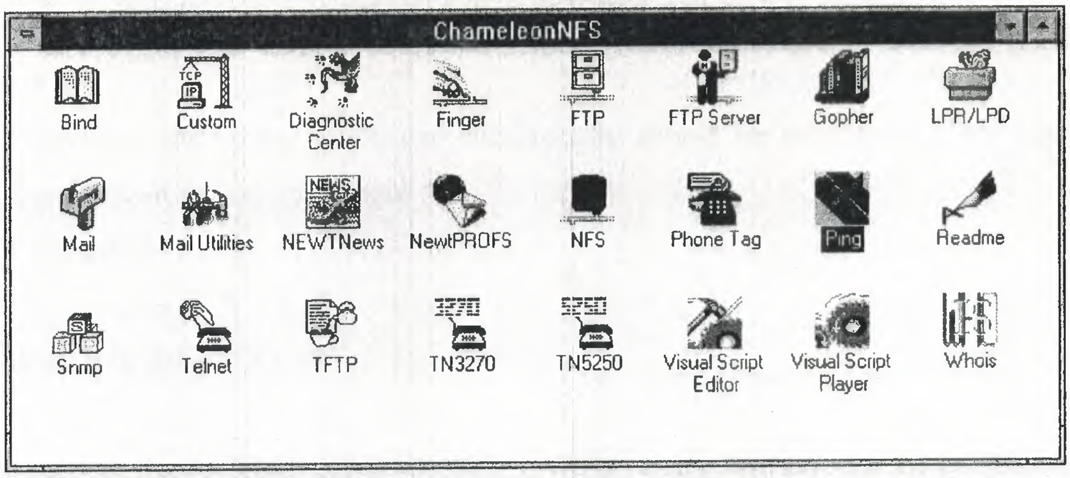


Figure 3.6 The Chameleon program group.

Because of the different installation variables encountered with different network drivers, it is advisable to check the following configuration files manually:

AUTOEXEC.BAT

CONFIG.SYS

PROTOCOL.INI

SYSTEM.INI

The following sections discuss each of these files in more detail. If the files do not have the information specified in them, add them with a text editor. Failure to check the files properly can result in Windows being unable to boot properly. If this happens, copy the backup files in place of the newly modified files, restart Windows, and reinstall or reconfigure as necessary.

3.2.2.1. The AUTOEXEC.BAT File

The changes to the AUTOEXEC.BAT file necessary to enable Chameleon to run are the inclusion of the installation directory in the PATH environment variable and a network startup command. If Chameleon is installed on a Windows for Workgroups system, the network startup command should already exist.

The PATH environment variable must be modified to include the Chameleon installation directory, which by default is C:\NETMANAG. An existing PATH statement can be altered, or a new line can be added below the existing PATH statement that looks like this:

```
PATH=C:\NETMANAG;%PATH%
```

Of course, the correct drive and subdirectory should be substituted. This chapter assumes default values throughout.
The command

```
C:\WINDOWS\NET START
```

is already in the AUTOEXEC.BAT file if a Windows for Workgroups system is used If Chameleon is installed on a Windows (not Windows for Workgroups) system, the NETBIND command included with the distribution software should be called as well:

```
C:\NETMANAG\NETBIND
```

Chameleon might install a SHARE command in the AUTOEXEC.BAT file if one does not exist. If one doesn't exist, it is advisable to add it if others can access the machine. SHARE is a DOS utility that activates file-sharing and record-locking. If other machines will be accessing the machine, SHARE is necessary to prevent error messages and potential system freezes when file conflicts occur.
The completed AUTOEXEC.BAT file looks like this for a Windows for Workgroups installation:

```
PATH=C:\NETMANAG;%PATH%
C:\WINDOWS\NET START
SHARE
```

and like this for a Windows installation:

```
PATH=C:\NETMANAG;%PATH%
C:\NETMANAG\NETBIND
SHARE
```

If the NET START or NETBIND command is not executed properly, Windows displays an error message when it loads. In some cases, Windows can lock up while it tries to access the network drivers.

3.2.2.2 The CONFIG.SYS File

The CONFIG.SYS file might be considerably different for each installation. The HIMEM memory device driver is required, and the SMARTDRIVE caching system is recommended. All installations should have adequate values for the FILES and BUFFERS settings, which are normally set by Windows when it is installed. The CONFIG.SYS should have these values as a minimum:

```
BUFFERS=30
FILES=30
LASTDRIVE=Z
STACKS=9,256
```

This creates enough file and buffer settings to enable multiple files to be open at once. Higher values are better, although there is a trade-off of efficiency once the values exceed a certain value (depending on the amount of RAM in a system). The LASTDRIVE setting enables more drives to be open than are physically connected to the system. This is necessary when remote drives are mounted, either through Windows for Workgroups or Chameleon.

For a Windows or Windows for Workgroups system, Chameleon adds the following commands to the CONFIG.SYS file:

```
DEVICE=C:\NETMANAG\PROTMAN.DOS /I:C:\NETMANAG
DEVICE=C:\NETMANAG\EXP16.DOS
DEVICE=C:\NETMANAG\NETMANAG.DOS
```

These load the device drivers for the protocol manager, the network interface card, and the specific protocol for Chameleon. The protocol manager and network interface card device drivers were discussed in the DOS section earlier today.

Windows for Workgroups usually has a command in the CONFIG.SYS file that looks like this:

```
DEVICE=C:\WINDOWS\IFSHLP.SYS
```

This automatically loads all the necessary drivers. In some cases, Chameleon adds the command for the Windows for Workgroups 3.1 device drivers to the end of the CONFIG.SYS file, even if the IFSHLP.SYS driver exists. Comment out the added device drivers and try the system without them. The IFSHLP.SYS device driver should be sufficient.

3.2.2.3 The SYSTEM.INI File

The Windows SYSTEM.INI file requires a few changes to ensure that Chameleon is loaded properly. These should be effected by the installation script, but check the lines carefully anyway.

The [boot] section of the SYSTEM.INI file should have the following two lines:

```
[boot]
shell=progman.exe
network.drv=C:\NETMANAG\MULT400.DRV
```

The shell line might be different if the system uses a replacement program manager (such as Central Point PC Tools for Windows Desktop Manager). The MULT400 driver supports several networks at a time. The order of these lines in the SYSTEM.INI file is not important, as long as they appear in the proper section. The MULT400 driver takes care of loading all the necessary drivers for each network. Windows for Workgroups should have this line

```
network.drv=wfwnet.drv
```

either commented out with a semicolon at the start of the line or removed entirely. The WFWNET driver is the Windows for Workgroups network driver, which must be replaced by MULT400.

The [boot.description] section of the SYSTEM.INI file is changed to

```
[boot.description]
network.drv=NetManage ChameleonNFS
```

or a similar line if another NetManage product is installed.
The [386Enh] section has several changes made. These are as follows:

```
[386Enh]
device=C:\netmanag\nmredir.386
network=*vnetbios,*vwc,vnetsup.386,vredir.386,vserver.386
netmisc=ndis.386,ndis2sup.386
netcard=
transport=nwlink.386,nwnblink.386,netbeui.386
InDOSPolling=FALSE
```

The order of the lines in the section doesn't matter. They load the correct network device drivers into the Windows kernel.
Finally, the [network drivers] section should have these lines:

```
[network drivers]
netcard=elnk3.dos
devdir=C:\WINDOWS
LoadRMDrivers=YES
transport=ndishlp.sys,c:\netmanag\netmanag.dos,*netbeui
```

The netcard line changes depending on the network interface card used. The LoadRMDrivers line should be changed from the Windows for Workgroups default value of NO to YES.

3.2.2.4 The PROTOCOL.INI File

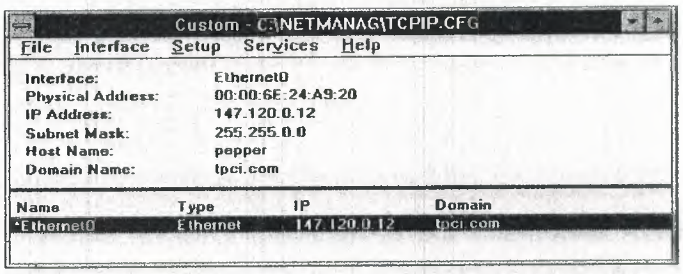
The PROTOCOL.INI file for a Windows for Workgroups installation doesn't require many changes. The driver information should already exist. A new section added by Chameleon should look like this:

```
[NETMANAGE]
DRIVERNAME=netmng$
BINDINGS=MS$ELNK3
```

The BINDINGS line changes depending on the network interface card. It is easiest to copy the line from another section of the PROTOCOL.INI file.

3.2.3 Configuring Chameleon

Once Chameleon has been installed and the startup files checked for proper content, you can configure the software for the sample machine. This is done through the Chameleon CUSTOM application. When started, CUSTOM displays a status screen as shown in Figure 3.7


Custom - C:\NETMANAG\TCPIP.CFG
File Interface Setup Services Help

```
Interface:          Ethernet0
Physical Address:   00:00:6E:24:A9:20
IP Address:         147.120.0.12
Subnet Mask:        255.255.0.0
Host Name:          pepper
Domain Name:        tpci.com

Name           Type        IP              Domain
*Ethernet0     Ethernet    147.120.0.12    tpci.com
```

Figure 3.7 The Chameleon Custom screen.

If the installation routine didn't add the machine's name and IP address to the Custom screen, use the Setup menu item to select the different aspects of the configuration that

must be specified. You should provide a machine name, IP address, subnet mask, and domain name, as well as the interface if not already added (Ethernet, in this case).
To enter the names of the other machines on the network and their IP addresses, select the Services menu Host Table option to display the Host Table dialog box. To add the other machines on the sample network, enter a name in the top portion of the window in the field titled Official Name and click the Add button. This shows a window for the IP address, which should be filled in completely. Then click OK. The IP address and the machine name are now entered into the host table. This window is shown in Figure 3.8 with the address for the machine merlin added. If a machine has more than one name, the different names can be added as aliases through this screen, as well.

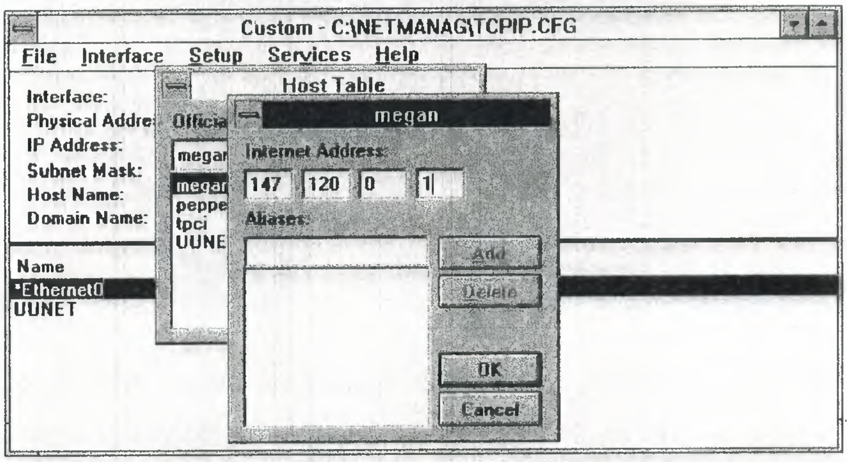
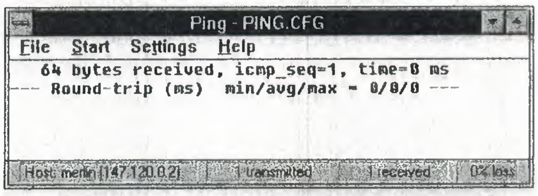


Figure 3.8 Chameleon's Host Table IP Address dialog box.

3.2.4 Testing Chameleon

After the changes to the four configuration files are completed, reboot the system and start Windows. Watch for error messages as the Chameleon lines in the CONFIG.SYS and AUTOEXEC.BAT files are executed. If Windows for Workgroups was installed and working prior to installing Chameleon, there should not be any errors.
The easiest way to test the new TCP/IP system is to use the ping utility within the Chameleon program group. When selected, it displays a small dialog box. Select the Start option, which displays another dialog box waiting for a machine name. Enter the name of the local machine.

The ping window should show a successful result. This is indicated by a message showing the number of bytes received, as well as time information. A sample output from a successful attempt to ping the local machine is shown in Figure 3.9

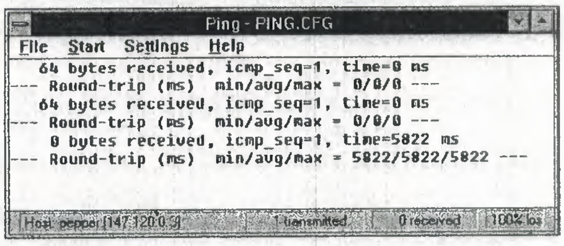


```
   64 bytes received, icmp_seq=1, time=0 ms
--- Round-trip (ms)  min/avg/max = 0/0/0 ---
```

Figure 3.9 ping diagnostic messages.

If the ping attempt is not successful, Chameleon displays a message about the network drivers not installed or about unreachable hosts. Upon receipt of such a message, check the network card settings and all the configuration information through the CUSTOM program.

The next step is to use ping to send to another machine on the network. Figure 3.10 shows the output from a ping attempt on freya, the sample network's Linux server and to whitney, the Windows machine that is not booted (and hence should fail). The system timed out on the whitney attempt, as you would expect.



```
   64 bytes received, icmp_seq=1, time=0 ms
--- Round-trip (ms)  min/avg/max = 0/0/0 ---
   64 bytes received, icmp_seq=1, time=0 ms
--- Round-trip (ms)  min/avg/max = 0/0/0 ---
    0 bytes received, icmp_seq=1, time=5822 ms
--- Round-trip (ms)  min/avg/max = 5822/5822/5822 ---
```

```
Host: pepper [147.120.0.3]    1 transmitted    0 received    100% los
```

Figure 3.10 Ping across a network.

If the ping attempts across the network fail on all machines, the problem is likely with the configuration. Check all the configuration information , as well as the network cables and cards. Make sure the machines to be pinged are up and running TCP/IP.

If the network is operating properly, try the ftp and telnet applications from the Chameleon program group. Full instructions for these utilities are in the documentation. As long as a host table entry has been created and ping succeeded, the other utilities should function properly. Both provide a graphical interface that Windows users will find familiar, instead of the character-based line interface found with DOS.

4. TROUBLESHOOTING TCP/IP

4.1 Troubleshooting the Network Interface

The physical connection to the network is a suitable starting point for troubleshooting when a problem is not obvious. Because there are many popular network interfaces, each of which must be dealt with in a slightly different manner, some generalizations must be made. The overall approach remains the same, however.
Assuming that the network itself is functional, the most common problems with the network interface are a faulty network card or a bad connector. Checking each is easily done by simple replacement. If the problem persists, the fault is most likely higher in the architecture.

Faulty network transport media (usually cables) are not uncommon. If a device at the end of a cable is not functioning, it is worthwhile to check the cable itself to ensure that a communication path exists. This can be done with a portable computer or terminal, or in some cases a conductivity tester, depending on the network. A systematic testing process can narrow down a network cabling problem to a specific segment.

One overlooked problem arises not because of a real fault with the network interface or the network itself, but because one device on the network is transmitting a different protocol. This can foul up the entire network and grind it to a halt. (For example, an Ethernet network might have one or more devices set to transmit IEEE 802.3 frames, which are not the same as Ethernet.)

If there is a conversion from one protocol to another, that can be suspect. For example, it is common to find AppleTalk networks running TCP/IP. The IP messages are encapsulated in AppleTalk frames. If the conversion between the two formats (which can occur at a gateway or router) is not clean, some faulty packets might be passed. This can cause network problems.

If the network connections and network interface cards appear to be working (which can be verified with a network analyzer or board swapping), the problem is in a higher layer.

4.2 Troubleshooting the Network (IP) Layer

The network layer (where IP resides) can be the most trouble-prone aspect of the network if configuration rules are not followed scrupulously. Because this layer handles routing, any mistakes can cause lost packets, making it appear that a machine on the network is not communicating with the others. ICMP can be a useful tool for troubleshooting this layer.

One of the most common mistakes, especially with large networks, is a duplication of IP addresses. This can be an accident, as a new address is programmed, or a user can move his or her machine and in the process jumble the IP address. It is not uncommon for users to change the IP address by mistake when investigating the software. The network mask must also be correct.

Addressing of packets within the IP layer (where the source and destination IP addresses are encapsulated in the IP header) is another source of problems. Determining destination IP addresses requires communications with another machine, which should hold the necessary information. If the Domain Name System (DNS) is active, it can contribute to the confusion if the server has faulty tables.

It is necessary for the IP address to be mapped to the physical address. Both ARP and RARP require this table to direct packets over the network. If a network card is changed for any reason, the unique physical address on the board no longer corresponds to the IP address, so messages are rerouted elsewhere. Network administrators must keep close track of any changes to the network hardware in all devices.

Problems can also occur with devices that handle intermediary routing, such as bridges, routers, and brouters. These must be aware of all changes to the network, as well as physical and logical addresses for the devices they are connected to. Specialized protocols such as Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) handle much of this maintenance, but somewhere in the network a manual notation of changes must be made.

There are many potential sources of trouble with the network layer. Even processes that should work without trouble, such as packet fragmentation and reassembly, can cause problems.

Connectivity between machines at both the transport and network level can be tested using utilities such as ping. A systematic check of machines along a network and out over an internetwork can help isolate problems, not just in the source and destination machines but also in intermediate processors such as routers. The traceroute utility can be used for this, also, if it is available.

4.3 Troubleshooting TCP and UDP

Assuming the network layer is functioning correctly, the host-to-host software might be a problem. If the software is correctly installed and started (which might sound obvious but is a common cause of failure), a process to isolate the problem must be followed. There are many files involved with both TCP and UDP, differing with each operating system version, so the documentation accompanying the TCP or UDP software should be consulted.

The protocol in use must be determined first: Is the machine using TCP or UDP, and if both, are both failing? Problems such as too many retransmissions or no timeout values can make UDP appear as if it is failing, but TCP would not be affected (unless it uses the same port or too many processes are active).
Port addresses can be problematic, especially with TCP. Each port on a machine can be sent a ping message from a remote machine to verify that it is communicating properly. If a port request fails, it might indicate an improper or missing entry in a configuration file. The finger utility might also be useful. If messages are passing correctly from one machine to another, the problem is in the configuration of the software, or a higher level application.

Incorrect configuration parameters can cause TCP or UDP failures. For example, if the send and receive window values for TCP are set to low levels, there might be no opportunity for applications to pass enough information. In this case, it might appear that TCP is at fault. Carefully check all configuration files and settings.

4.4 Troubleshooting the Application Layer

Assuming that both IP and TCP or UDP are functioning properly, the application layer is suspect. It is in this layer that higher-level protocols such as the File Transfer Protocol (FTP), Telnet, and SMTP are based. It can be difficult to find problems within the application layer, although a few simple tests help eliminate obvious solutions.

Several commercial utilities are available to monitor reception within the application layer.

Assuming that data is getting to the right application (which can be checked with some diagnostic tools or simple programming routines), the problem might be in interpretation. Verify that the communications between two applications are both the same format. More than one application has expected ASCII and received EBCDIC. Diagnostics show the messages moving into the application properly, but they are total gibberish to the application when it tries to interpret them.

Assuming that is not the problem, there could be a fault with the applications at either end. Although you might assume that a Telnet program from one vendor would talk to one from another vendor, this is not true in an unfortunately large number of cases. If there are no identical software packages or versions known to work with the other package, this can be difficult to troubleshoot. This kind of cross-application problem is particularly prevalent with mixed-platform systems, such as a PC-based FTP or TCP/IP software package trying to access services on a UNIX host.

Some readily available utilities can be used to monitor the application layer. Some of these utilities are distributed with operating systems, and others are distributed as public domain software. The utility snmpwatch is a network monitoring program that reports on any SNMP variables that change their values. This can be helpful in diagnosing communications problems within SNMP.

The Internet Rover is a network monitoring program that enables testing of several protocols, including Telnet, FTP, and SMTP. Unfortunately, it doesn't work with all operating system variants. Another tool for SMTP testing is mconnect, which verifies connections.

4.5 Security

Security is an important issue and one often overlooked, usually to the administrator's. Taking the steps to set up a proper security policy and protecting the system as well as possible should be a mandatory task for every system administrator.
Routers can be significant in a network's security plan. Most routers enable the system administrator to restrict traffic through the router in some manner, either in one direction or both. A router can be set, for example, to prohibit Telnet or rlogin requests from outside the network, but enable through file transfer requests such as FTP. Routers can

also prevent traffic into a local network through the router from anywhere outside the network, cutting down on access into (and through) a network.

Routers usually perform this type of traffic filtering by simply looking at the datagram headers for the requested port. If one of the restricted ports is requested, the datagram can be returned to the sender or discarded. Setting the proper access filters from a network router can be an effective and simple manner of restricting outside access.

Unfortunately, the Internet and most networks were simply not designed to prevent unauthorized access or monitoring. These features were usually added as an afterthought, and as such have some problems. Watching network traffic and trapping addresses, user IDs, and passwords is ridiculously easy, so MIT developed Kerberos security protocols to help.

Kerberos (named after the three-headed dog guarding the gates of Hades) uses an encryption key and server introduction method to enable access. Kerberos is slowly being adopted as a standard among Internet users (despite some governmental protests), and it works well with the TCP/IP family of protocols.

CONCLUSION

Internet Protocols are the standard, routable entries networking protocols. All modern operating systems offer TCP/IP support and most large networks rely on TCP/IP for much of their network traffic. This is a technology for connecting dissimilar systems. Many standard connectivity utilities are available to access and transfer data between dissimilar systems, including File Transfer Protocol and Telnet. It provides a robust, scalable, cross-platform client/server framework. TCP/IP offers the socket interface, which is ideal for developing client/server applications that can run on Sockets-compliant stacks from other venders. Sockets applications can also advantage of other networking protocols such as NWLink used in Novell Net Ware networks. Internet Protocols provide a method of gaining access to the Internet. The internet consists of thousands of network worldwide connecting research facilities, universities, libraries, government agencies and private companies.

REFERENCES

[1] TCP Extensions for Long-Delay Paths, Jacobson, V.; Braden, R.T.; 1988

[2] On the Assignment of Subnet Numbers, Tsuchiya, P.F.; 1991

[3] Remote Network Monitoring Management Information Base, Waldbusser, S.; 1991

[4] James Chellis, Charles Perkings, Matthew Strebe, "Networking Essentials", SYBEX Publishers 1999.

[5] James Chellis, "Windows 2000 Network Infra Structure" SYBEX Publishers 2000

[6] Charles W. "Understanding TCP/IP" BPB Publishers 1996

[7] http://www.us-epanorama.net

[8] http://www.microsoft.com

[9] http://www.commweb.com

[10] http://www.oreily.com

[11] http://www.cisco.com

[12] http://www.grouper.ieee.org